

ARBEITSBERICHTE DES INSTITUTS FÜR MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG (INFORMATIK)

FRIEDRICH ALEXANDER UNIVERSITÄT ERLANGEN NÜRNBERG

HERAUSGEBER: H. Billing, W. Händler, U. Herzog,
F. Hofmann, K. Leeb, P. Mertens, H. Müller, G. Nees, H. Niemann,
D. Seitzer, B. Schmidt, H. J. Schneider, H. Wedekind

LEISTUNGSANALYSE VON MULTIPROZESSORSYSTEMEN MIT PROZESSKOMMUNIKATION

Ian Fuat Akyildiz

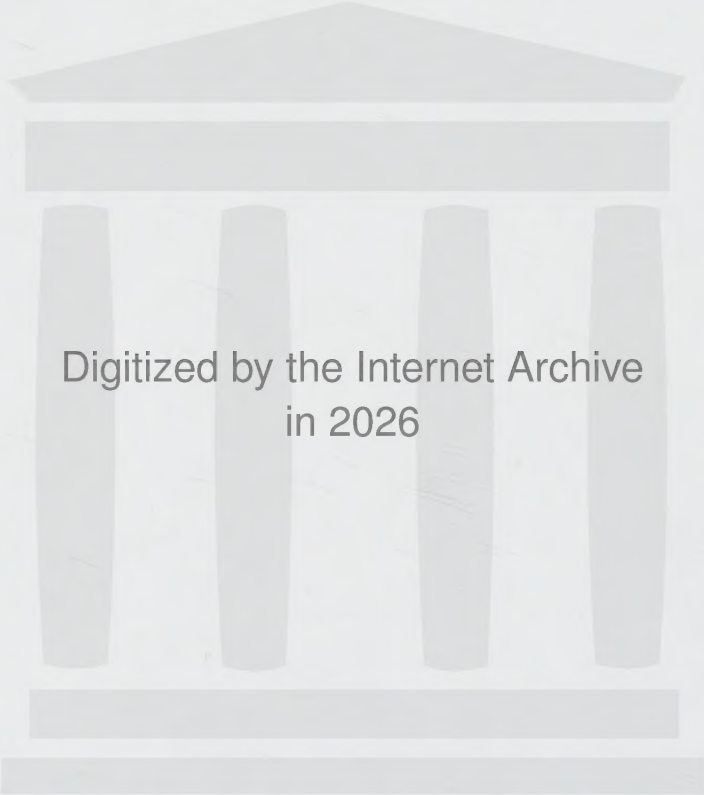
DISSERTATION

BAND 17 · NUMMER 8 · ERLANGEN · DEZEMBER 1984

1985 60



ISSN 0344 - 3515



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41546206_0023

LEISTUNGSANALYSE VON MULTIPROZESSORSYSTEMEN MIT PROZESSKOMMUNIKATION

Der Technischen Fakultät
der Universität Erlangen-Nürnberg
zur Erlangung des Grades
DOKTOR - INGENIEUR

vorgelegt

von

Ian Fuat Akyildiz

ERLANGEN 1984

Als Dissertation genehmigt von der
Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung 10.7.1984

Tag der Promotion 13.9.1984

Dekan Prof. Dipl.-Ing. M. Vetter

Berichterstatter 1. Prof. Dr.-Ing. U. Herzog
2. Prof. Dr. rer. nat. F. Hofmann

Für

Céline Rengin (geb. Okt. 1982) und

meine vielgeliebte Mutter (gest. März 1983)

10 Jahre Erlangen

Es ist fast 10 Jahre her, daß ich hierher gekommen bin, wo ich sehr viele Hoch- und Tiefpunkte erlebt habe. Bald werde ich diese Universität und Stadt verlassen, denen ich einen großen Teil meiner Jugend gegeben habe.

Als ob es heute passiert wäre, kann ich mich noch daran erinnern, wie ich im August 1974 vom Akademischen Auslandsamt zum damaligen Studienberater B. Schmidt geschickt worden war, den ich nicht antraf, weil er in Urlaub war. Bei der Suche nach ihm begegnete ich einem freundlichen Herrn, der mich in sein Büro führte und ausführlich über das Studium der Informatik an der Universität Erlangen-Nürnberg beriet. Beim Verlassen seines Büros fragte ich ihn, wer er eigentlich sei, worauf er auf sein Namensschild an der Tür deutete : "Prof. Dr. F. Hofmann". Wer hätte damals geahnt, daß ich 7 Jahre später an seinem Lehrstuhl tätig sein würde? Ich bin stolz darauf, seinem Lehrstuhl angehört zu haben.

Als ich im Jahre 1979 meine Studienarbeit über die Methoden zur Analyse von Warteschlangennetzen für Herrn Dr. Bolch schrieb, die später als Arbeitsbericht und als Buch veröffentlicht wurde, wollte ich auch die sehr bekannte Parametrische Analyse von Chandy, Herzog und Woo in die Arbeit aufnehmen. Für einige Fragen bin ich zu Prof. Herzog gegangen. Vor so einem bekannten Professor zu stehen: ich habe alles durcheinander gebracht und konnte vor lauter Aufregung nicht genau formulieren, was ich eigentlich wollte. Wer hätte damals gedacht, daß ich später bei Prof. Herzog promovieren würde, obwohl ich die erste Enttäuschung in meinem Arbeitsleben durch ihn erlebt hatte? Die Universität Erlangen muß stolz sein, einen solchen Professor in seinen Reihen zu haben.

Als ich im Jahre 1979 eine Studienarbeit suchte, ging ich zu Herrn Dr. Bolch, der damals aus Rio zurückgekommen war. Damit hat er in meiner Karriere eine große Rolle gespielt, weil er mich mit dem Gebiet der Warteschlangennetze bekannt gemacht hat. Daher werde ich ihn immer in dankbarer Erinnerung behalten.

Die Herrn W. Schneider, Dipl.-Inf. T. Stroup und Dipl.-Inf. M. Paterok, die meine Launen ausgehalten haben und mich psychisch unterstützt und aufgebaut haben, werde ich nie vergessen. Ich bin

sehr froh, daß ich in Herrn Paterok endlich jemanden gefunden habe, der sich auch mit Leib und Seele mit Warteschlangennetzen beschäftigt. Bei der Vervollständigung meiner Arbeit hat er mich sehr unterstützt. Ich hoffe, daß ich später so begeistert mit ihm weiterarbeiten kann. Ich muß noch viele meiner Studenten erwähnen, die mich sowohl bei den Untersuchungen als auch beim Tippen unterstützt hatten.

Ich wäre in meinem Leben nie so weit gekommen, wenn ich meine lieben, schon verstorbenen Eltern, meine liebe Frau Maria und meinen eisernen Willen nicht gehabt hätte.

ZUSAMMENFASSUNG

In dieser Arbeit wird ein Multiprozessorsystem, in dem Prozesse über Puffer kommunizieren, analytisch untersucht. Aufgrund endlicher Pufferkapazitäten können bei der Kommunikation Blockierungen vorkommen, wie es z.B. beim iAPX-432 Multiprozessorsystem der Fall ist.

Die Analyse erfolgt hierarchisch anhand zweier Modelle (Gesamt- und Software-Modell), wobei im Software-Modell, bestehend aus Stationen mit endlichen Warteschlangenkapazitäten die Prozeßkommunikation untersucht wird. Mit Hilfe dreier Methoden, die in dieser Arbeit vorgestellt werden und auf Netze mit beschränkten Stationskapazitäten anwendbar sind, werden die Blockierzeiten und Blockierwahrscheinlichkeiten der einzelnen Prozesse ermittelt. Eine der Methoden liefert für Tandem-Netze exakte Ergebnisse.

Die Ergebnisse aus dem Software-Modell werden als Eingabeparameter für das Gesamtmodell verwendet und die Leistungsgrößen für das Multiprozessorsystem berechnet. Für die Analyse des Gesamtmodells werden verschiedene Algorithmen vorgeschlagen, die teils Erweiterungen existierender Methoden sind.

Für die Validierung der analytischen Ergebnisse wird das Multiprozessorsystem simuliert.

INHALTSVERZEICHNIS

1. Einführung	1
1.1 Methoden zur Leistungsanalyse	1
1.2 Der Modellierungsprozeß	4
1.3 Übersicht über die Warteschlangennetzwerktheorie	6
1.4 Übersicht über diese Arbeit	15
2. Beschreibung des Multiprozessorsystems (Systemmodell)	18
3. Modellierung des Multiprozessorsystems	22
3.1 Gesamtmodell	23
3.2 Software-Modell	25
4. Algorithmen für die Analyse des Software Modells	28
4.1 Zustandsraumtransformationmethode für geschlossene Tandem-Netze mit Blockierung	32
4.2 Zustandsraumtransformationmethode für allgemeine geschlossene Warteschlangennetze mit Blockierung	50
4.3 Mittelwertanalyse für Blockiernetze (MVABLO)	66
4.4 Produktformlösungen für geschlossene Warteschlangen- netze mit Blockierung	75
5. Analyse des Gesamtmodells	85
5.1 Beispiel 1	89
5.2 Beispiel 2	96
5.3 Beispiel 3	101
6. Zusammenfassung und Vorschläge für die Zukunft	106

ANHANG I

A1. Algorithmen für die Analyse des Gesamtmodells	112
A1.1 Warteschlangennetze mit Produktformlösungen	112
A1.1.1 TUPAL (Throughput Algorithm)	116
A1.1.2 Approximation der Mittelwertanalyse für lastabhängige Stationen	125
A1.1.3 Reduktionsmethode	136
A1.2 Nichtproduktform-Netze	149
A1.2.1 Erweiterte Mittelwertanalyse (EMVA)	150
A1.2.2 SIMVAG- Algorithmus	152
ANHANG II. Simulation des Gesamtsystems	156
ANHANG III. Ergebnisse für Netze mit Blockierung	166
Symbolverzeichnis	171
Literaturverzeichnis	172

1. EINFÜHRUNG

Die Leistungsanalyse von Rechensystemen erfolgt im wesentlichen aus 3 Gründen:

- aus einer Anzahl bereits bestehender Rechensysteme sollen diejenigen ausgesucht werden, die für die eigenen Zwecke am geeignetsten sind.
- die Leistungsfähigkeit eines noch nicht bestehenden Systems soll vorhergesagt werden, um die Effizienz neuer Konzepte und Strukturen des Entwurfs zu untersuchen.
- man versucht, Engpässe in einem Rechensystem zu bestimmen.

Die Leistungsanalyse von Rechensystemen wird durch Meßmethoden oder Modellierungstechniken durchgeführt.

Bei existierenden Systemen kann die Leistung direkt durch Meßmonitore (Hard-/Software Monitore) ermittelt werden. Obwohl die Messung am realen System ein wichtiger Bestandteil jeder Leistungsanalyse ist, stehen einer ausschließlichen Verwendung von Meßtechniken Probleme entgegen; wie z.B. daß in der Entwurfs- und Entwicklungsphase eines Systems Messungen nicht durchführbar sind.

Da zudem das Ablaufgeschehen in heutigen Rechensystemen zu komplex ist, um nur durch Messungen erfaßt oder vorhergesagt zu werden, werden zur Analyse Modelle verwendet. Anhand dieser Modelle kann das dynamische Ablaufgeschehen in Rechensystemen bewertet und optimiert werden. Damit können reale Abläufe formal beschrieben, charakteristische Leistungsgrößen wie z.B. Auslastung der Geräte, Durchsatz des Systems, Verweil- und Wartezeiten der Prozesse in den einzelnen Geräten usw. definiert und bestimmt und Entscheidungshilfen für den Entwurf optimaler Hardwarestruktur und Betriebsprogramme bereitgestellt werden.

Diese Modelle stellen nur die für die spezielle Analyse relevanten Merkmale des Systems dar, wie z.B. wichtige Systemkomponenten oder Beziehungen und Datenverkehr zwischen diesen Komponenten. Die Systemmerkmale lassen sich unter 3 Gesichtspunkten ordnen, wobei jeweils einige typische Beispiele angegeben werden:

i) Hardware

- Anzahl der Prozessoren und deren Rechengeschwindigkeit
- Anzahl und Abarbeitungsgeschwindigkeit der peripheren Geräte.
- Größe des Hauptspeichers
- Anzahl von Blöcken, Ankunftsabständen und Blocklängen
- Ausgewogene Dimensionierung der Pufferspeicher

ii) Benutzer des Rechensystems

- Multiprogramming-Grad d.h. die Anzahl der interaktiv arbeitenden Benutzer
- Prozessorzeit eines Auftrags und die Anzahl seiner E/A-Operationen

iii) Betriebssystemfunktionen

- Abarbeitungsstrategien für die Aufträge in Warteschlangen
- Speicherzuteilungsstrategien
- Anzahl der Seitenzugriffe pro Zeiteinheit
- Zuordnung von Dateien zu peripheren Geräten
- Prozentualer Anteil der Rechenzeit für Verwaltung (Overhead), Datentransfer
- Grad der Parallelität einzelner Benutzerprogramme insbesondere mittlere und maximale Anzahl gleichzeitig bearbeitender Teilprozesse, Synchronisation zwischen dieser Teilprozesse.

Für die Modellierung dieser Charakteristika eignen sich die Warteschlangennetze (queueing networks), die in den letzten Jahren für die Leistungsanalyse von Rechensystemen immer häufiger angewandt werden. Für andere Zwecke werden geeignete andere Modellkonzepte wie z.B. Graph-Modelle (insbesondere Petri Netze) für theoretische Untersuchungen, Diagnosegraphen für Zuverlässigkeitsuntersuchungen und Netzwerkflußmodelle für Kapazitätsüberlegungen verwendet.

Die Warteschlangennetzmodelle können auf unterschiedlichem Wege untersucht werden:

- a) Analytische Methoden
- b) Simulation
- c) Hybrid-Simulation

Bei der analytischen Vorgehensweise wird versucht, auf mathematischem Wege Beziehungen zwischen fundamentalen Systemgrößen wie z.B. Anzahl der Geräte, Anzahl der Prozesse in den einzelnen Geräten usw. und relevanten Leistungsgrößen wie z.B. Auslastung, Durchsatz, Antwortzeit, Wartezeit, usw. herzuleiten.

Die Vorteile der analytischen Methoden sind zum einen die leicht programmierbaren und schnellen Algorithmen, zum anderen die meist absolute Abhängigkeit zwischen Modellparametern und Leistungsgrößen. Der Nachteil der analytischen Methoden ist, daß sie begrenzte Anwendungsmöglichkeiten haben.

Bei der Simulation werden die Vorgänge im Rechner mit speziellen Computer-Programmen nachgespielt. Da das Verhalten eines Simulationsmodells in Bezug auf die relevanten Parameter dem Verhalten des realen Systems entspricht, können daraus alle zur Leistungsbewertung interessanten Größen ermittelt werden. Nachteile der Simulation sind die zeit- und kostenaufwendige Vorbereitung und Ausführung eines Modells sowie die schwer erkennbare Abhängigkeit der Parameter und die daraus resultierende umständliche Optimierung.

Die Hybrid-Simulation ist eine Kombination der analytischen Methoden mit der Simulation. Dabei werden Teilsysteme mit analytischen Methoden analysiert und diese Werte dann bei der Simulation mitverwendet oder umgekehrt.

1.1 DER MODELLIERUNGSPROZESS

Der Prozeß zur Erstellung der Warteschlangennetzmodelle wird hierarchisch mit abnehmender Komplexität durchgeführt. (Fig.1.1)

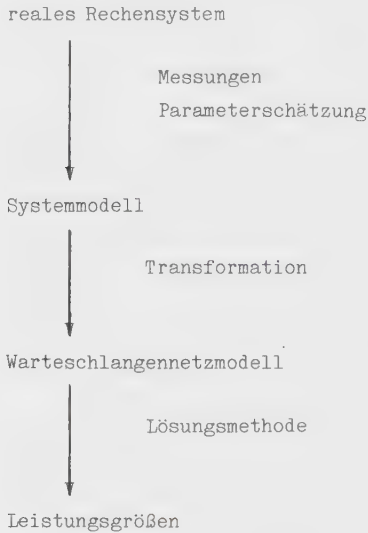


Fig. 1.1 Der Modellierungsprozeß

Dabei unterscheiden wir zwischen System- und Warteschlangennetzmodell.

Das Systemmodell

stellt alle wesentlichen Teile und Funktionen eines realen Rechensystems dar. Dieses Modell wird entwickelt, um das komplexe Verhalten des realen Systems zu erfassen. Zur Aufstellung dieses Modells braucht man gute Kenntnisse über das reale Rechensystem, seine Hardware, Software und die Last. Die Systemparameter müssen mit Hardware- und Softwaremonitoren gemessen und die Parameter, die nicht gemessen werden können, müssen abgeschätzt werden.

Das Warteschlangennetzmodell

Aus dem Systemmodell wird das Warteschlangennetzmodell konstruiert, wobei die detaillierten Systemmodellparameter in weniger detaillierte Eingabeparameter für das Warteschlangennetzmodell transformiert werden müssen, um die Voraussetzungen zu erfüllen, die die vorhandenen analytischen Lösungsmethoden an das Modell stellt.

Falls keine analytische Methoden für das entwickelte Warteschlangennetzmodell existieren, müssen dann entweder neue Methoden entwickelt werden oder das Warteschlangennetzmodell noch mehr abstrahiert werden, damit es mit den bereits existierenden Methoden analysiert werden kann. Dadurch können dann die gewünschten Leistungsgrößen aus den Eingabeparametern berechnet werden.

Bei der Modellierung mit Warteschlangennetzen unterscheiden wir zwischen 3 Phasen :

- i) In der Entwurfsphase wird das Warteschlangennetzmodell erstellt. Anhand verschiedener analytischen Methoden werden dann die Leistungsgrößen aus den eingegebenen Systemparametern ermittelt.
- ii) In der Validierungsphase werden entweder das Warteschlangennetzmodell simuliert oder Werte aller Systemparameter am realen Rechensystem gemessen, z.B. die mittlere Bearbeitungszeit eines Jobs in der CPU, die mittlere Anzahl von E/A-Operationen pro Job usw. Die analytisch erhaltenen Leistungsgrößen werden mit simulativen oder gemessenen Werten verglichen. Die Güte des Modells hängt davon ab, wie gut analytischen mit simulativen oder gemessenen Werten für die Leistungsgrößen übereinstimmen. Bei mangelnder Übereinstimmung müssen die Entscheidungen der Entwurfsphase überprüft werden.
- iii) In der Vorhersagephase kann das Leistungsverhalten des Systems mit einem gültigen Modell unter veränderten Bedingungen an Hardware oder Software (z.B. Erhöhung der Prozessorzahl, Vergrößerung des Speicherplatzes) bestimmt werden.

1.2 ÜBERSICHT ÜBER DIE WARTESCHLANGENNETZWERKTHEORIE

Die ersten Untersuchungen von Rechensystemen mit Warteschlangenmodellen beschränken sich auf Modelle mit nur einer Bedienstation (Warteschlange + Bedieneinheit). Man betrachtete lediglich den Prozessor als wichtigste Komponente des Systems oder das gesamte System als eine einzige Bedienstation.

Warteschlangennetze, die aus mehreren Bedienstationen bestehen, werden dem wirklichen Aufbau von heutigen Rechensystemen mit vielen Komponenten besser gerecht.

Dabei wird zwischen offenen, geschlossenen und gemischten Warteschlangennetzen unterschieden. Ein Warteschlangennetz ist offen, wenn Ankünfte von Aufträgen von außen und Abgänge von Aufträgen nach außen bei jeder Station möglich sind. (Fig.1.2)

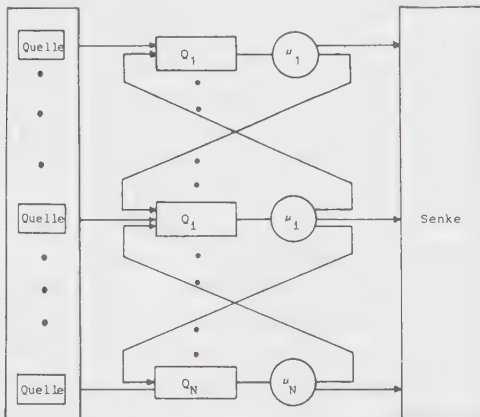


Fig. 1.2 Offenes Warteschlangennetz

Offene Modelle werden häufig für Rechnernetze angewandt.

Ein Warteschlangennetz ist geschlossen, wenn die Anzahl der Aufträge im System konstant ist, d.h.: Wenn ein Auftrag das System verlassen hat, rückt sofort ein neuer nach, so daß die Anzahl der Aufträge im System immer konstant bleibt. (Fig.1.3)

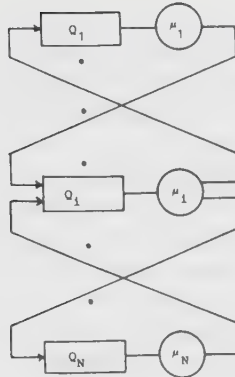


Fig. 1.3 Geschlossenes Warteschlangennetz

Mit geschlossenen Warteschlangennetzen kann man das Verhalten vieler Rechensysteme besser modellieren.

Ein Warteschlangennetz ist gemischt, wenn es aus offenen und geschlossenen Teilketten besteht. Ein Rechensystem mit On-line und Batch-Betrieb kann z.B. als ein gemischtes Netz modelliert werden. (Fig.1.4). Der On-Line-Betrieb wird durch die geschlossene Kette (die Anzahl der Benutzer ist konstant), der Batch-Betrieb durch die offene Kette modelliert.

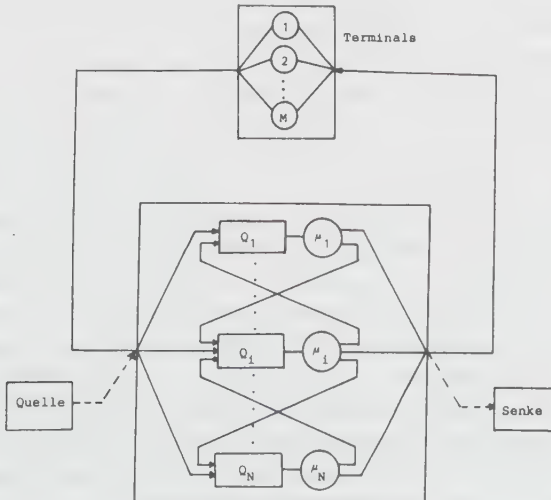


Fig. 1.4 Gemischtes Warteschlangennetz

Warteschlangennetze können auf unterschiedlichem Wege analytisch untersucht werden. Im folgenden werde ich diese Vorgehensweisen klassifizieren und einen Literaturüberblick geben.

Exakte Analyse für Produktformnetze

Die grundlegenden Ergebnisse der analytischen Modellbildung auf der Basis von Warteschlangennetzen sind in /JACK 57,63/ und /GORD 67/ angegeben. In /JACK 63/ werden Modelle mit mehreren Stationen behandelt, wobei jede Station eine oder mehrere Bedieneinheiten enthalten kann, die Bedienzeiten exponentiell verteilt sind und bei jeder Station neue Aufträge ankommen oder weggehen können. Als wichtiges Ergebnis für die Theorie der exponentiellen Warteschlangennetzwerke wird gezeigt, daß die Lösungen für die Zustandswahrscheinlichkeiten dieser Netzwerke Produktform haben, d.h. die Resultate sind Produkte aus Termen, die sich auf die einzelnen Stationen beziehen. Eine Vereinfachung dieses Ergebnisses für geschlossene Netzwerke durch geeignet gewählte Notationen wird in /GORD 67/ angegeben. Obwohl diese Produktformlösungen formelmäßig sehr einfach sind, erfordern sie doch einen erheblichen Rechenaufwand, wenn für ein konkretes System Leistungsgrößen berechnet werden sollen. Daher wurden in /BUZE 71/ Algorithmen eingeführt, die zu einer erheblichen Verminderung des Rechenaufwandes führen. Außerdem wird dort das sogenannte "Central-Server-Modell" vorgestellt, das sich besonders gut für Rechensysteme mit Multiprogramming eignet. Central-Server-Modelle haben auch dann noch Produktformlösungen, wenn die Bedienzeitverteilung bei der CPU eine rationale Laplace-Transformierte hat und gleichzeitig die Jobs mit PS (Processor Sharing) abgearbeitet werden. In /CHAN 72/ werden diese Ergebnisse auf beliebige Netze erweitert, alle Stationen können beliebige Bedienzeitverteilung haben, wenn die Warteschlangendisziplin PS oder LCFS (Last-Come-First-Served) ist. In /BCMP 75/ werden die obigen Ergebnisse verwendet, um Produktformlösungen für Warteschlangennetze mit mehreren Jobklassen, nichtexponentiellen Bedienzeitverteilungen und verschiedenen Warteschlangendisziplinen zu erhalten. Weitere Verbesserungen dieser Methode durch effizientere Algorithmen werden in /WONG 75, REIS 75/ angegeben.

/BUZE 71/ war die entscheidende Arbeit für die Anwendung von Warteschlangennetzen auf reale Rechensysteme. Seither gab es große Fortschritte in der Entwicklung von geeigneten Methoden für die Anwendung realer Systeme. Da sich bei Verwendung der Normalisierungskonstanten Over-/Underflowprobleme ergeben, hat man versucht, Leistungsgrößen ohne diese Größe zu bestimmen. Dabei entstand die Mittelwertanalyse (Mean Value Analysis), in Kurzform MVA, /REIS 78,79,81/, mit der geschlossene Warteschlangennetze mit Produktformlösungen unter ausschließlicher Verwendung von drei Gleichungen für die mittlere Verweilzeit, den Durchsatz und die mittlere Anzahl von Aufträgen in den Stationen untersucht werden.

/SAUE 81/ haben aus der Mittelwertanalyse einen Algorithmus LBANC (Local Balance Algorithm for Normalizing Constant) abgeleitet, mit dem man aus der dabei iterativ bestimmten Normalisierungskonstanten die Leistungsgrößen berechnet.

Numerische Methoden

Wenn ein Warteschlangennetz die Bedingungen für Produktformlösungen nicht erfüllt, weil z.B. eine Station eine hyperexponentielle oder Erlang-Bedienzeitverteilung und FCFS-Abarbeitungsstrategie hat, können wir die oben erwähnten effizienten Algorithmen nicht anwenden, um richtige Ergebnisse zu erzielen. In solchen Fällen können die iterative numerische Methode /STEW 78/, die direkte numerische Methode /STEW 78/ oder die rekursive numerische Lösungsmethode /HERZ 75, SAUE 81/ angewandt werden.

Die numerische Analyse wird wie folgt durchgeführt:

Zunächst müssen alle möglichen Zustände und die Übergangsraten zwischen diesen Zuständen im Modell gefunden werden. Damit wird die Matrix der Übergangsraten aufgestellt. Ein beliebiger Anfangsvektor der Zustandswahrscheinlichkeiten wird gewählt und mit den Iterationen $n=0,1,2,\dots$ begonnen. Die Iteration wird solange durchgeführt, bis sich die Werte bei der n -ten Iteration um weniger als ein geeignet gewähltes ϵ von denen der $(n-1)$ -ten Iteration unterscheiden. Die anderen Leistungsgrößen werden dann aus dem Ergebnis der n -ten Iteration berechnet.

Numerische Methoden besitzen einen prinzipiell uneingeschränk-

ten Anwendungsbereich. Außer der Forderung nach der Endlichkeit des Zustandsraumes unterliegen sie keinerlei Einschränkungen bzgl. der zu lösenden Modelle. Die numerische Analyse ist besonders geeignet zur Untersuchung von Warteschlangennetzen, die kleine Anzahl von Aufträgen und Bedienstationen haben. Ansonsten ist die Aufstellung der Matrix der Übergangsraten sehr zeit- und speicherplatzaufwendig, da der Zustandsraum für umfangreichere Systeme und bei einer großen Anzahl von Aufträgen sehr groß werden kann.

Approximative Methoden

Da die Produktformllösungen für eine beschränkte Klasse von Netzen existieren und die numerischen Methoden nur auf kleine Netze anwendbar sind, müssen wir approximative Methoden verwenden. Es gibt approximative Methoden, die auf der Produktform basieren. Dazu zählen die Diffusionsapproximation und die Erweiterte Produktformmethode.

Bei der Diffusionsapproximation wird der diskrete Prozeß "Anzahl der Aufträge in der i -ten Bedienstation zum Zeitpunkt t " auf einen kontinuierlichen Prozeß (Diffusionsprozeß) abgebildet und für die Schwankungen der Anzahl der Aufträge in einem Zeitintervall eine Normalverteilung angenommen. Die Wahrscheinlichkeitsverteilung der kontinuierlichen Prozesse (Diffusionsprozesse) kann durch die sogenannte Fokker-Planck-Gleichung beschrieben werden, die mit Hilfe geeigneter Randbedingungen gelöst wird. Man erhält die Dichtefunktion für den kontinuierlichen Prozeß und hieraus durch Integration approximierte Werte für die Zustandswahrscheinlichkeiten. Bei dieser Vorgehensweise können beliebige Ankunfts- und Bedienzeitverteilungen in den Stationen berücksichtigt werden. /KOBÄ 74, GELE 75/ haben die Diffusionsapproximation für allgemeine offene und geschlossene Netzwerke mit beliebiger Bedienzeitverteilung angewandt. Genauigkeitsuntersuchungen hierzu werden in /REIS 74/ durchgeführt. Die mit dieser Methode erzielten Ergebnisse liegen bei exponentiell angenommener Verteilung nahe den exakten Werten. Bei nichtexponentiellen Verteilungen sind die Abweichungen zu groß.

Eine andere Vorgehensweise der Approximation ergibt sich aus

der Zerlegung der Netze in Teilnetze. Solche Verfahren werden Aggregation-/Dekompositionsverfahren genannt. Dazu zählen die Iterative Approximation /CHAN 75a,b/, die Dekompositionsapproximation /COUR 77/, die Methode von Kühn /KÜHN 79/ und die Methode von Marie /MARI 80/.

/CHAN 75a/ haben Norton's Theorem aus der Theorie der elektrischen Netzwerke auf Warteschlangennetze angewandt. Ausgehend von einem allgemeinen Netz wird eine ausgewählte Station kurzgeschlossen, d.h. die Bedienzeit wird gleich Null gesetzt und dann werden die restlichen Stationen durch eine sogenannte zusammengesetzte (composite) Station ersetzt. Damit erhält man ein nur aus zwei Stationen bestehendes Netz, das einfacher zu analysieren ist. Diese Methode liefert für Produktformnetze exakte Lösungen. Die Erweiterung dieser Parametrischen Analyse zur iterativen Approximation für Systeme, die keine Produktformlösungen besitzen, wird in /CHAN 75b/ vorgeführt.

Die Grundidee der iterativen Approximation ist, daß das zu analysierende Netz A, das keine Produktformlösung liefert, durch ein Hilfsnetz B approximiert wird, das dieselbe Struktur mit denselben Bedienraten hat wie A und von dem vorausgesetzt wird, daß es Produktformlösungen liefert. Dieses Hilfsnetz B wird mit der Parametrischen Analyse auf zwei zyklisch verbundene Bedienstationen reduziert und dann mit der rekursiven Methode /HERZ 75/ analysiert. Die sich ergebenden Leistungsgrößen sind Näherungslösungen für das gegebene Netz A. Mit bestimmten Kriterien wird die Genauigkeit der Ergebnisse überprüft. Sind sie zu ungenau, dann wird das Hilfsnetz B mit gezielt geänderten Bedienraten erneut mit der beschriebenen Methode untersucht. Dieser Vorgang wird so oft wiederholt, bis die Genauigkeitskriterien eingehalten werden. Die sich dann ergebenden Leistungsgrößen stellen die endgültigen Näherungslösungen des Netzes A dar.

/MARI 80/ hat die iterative Approximationsmethode etwas modifiziert und in einer verfeinerten Version präsentiert. Die Methode von Marie gilt für geschlossene Warteschlangennetze, die einheitlich aus FCFS-Bedienstationen mit nichtexponentiellen Bedienzeitverteilungen bestehen. Diese Approximation basiert auf dem Konzept des bedingten Durchsatzes einer Bedienstation: bei einem gegebenen geschlossenen Warteschlangennetz wird jede Be-

dienststation einzeln unter einem Markov'schen Ankunftsprozeß mit lastabhängiger Ankunftsrate analysiert, d.h. als Lösung für Bedienstationen mit einer Bedieneinheit, FCFS-Strategie und allgemeinen Bedienzeitverteilungen (Cox-Verteilungen) verwendet man $\lambda(k)/G/1$ -Lösungen, mit deren Hilfe man den bedingten Durchsatz und daraus die Zustandswahrscheinlichkeiten berechnen kann. Hat man diese Zustandswahrscheinlichkeiten, so kann man daraus leicht alle anderen Leistungsgrößen bestimmen. Die Iterationen, bei denen jeweils nur die Ankunftsraten geändert werden, werden beendet, wenn die für die einzelnen Bedienstationen berechneten Ergebnisse die Durchsatz- und Auftragsinvarianz innerhalb einer vorgesehenen Fehlertoleranz erfüllen.

Bei der Dekompositionsapproximation /COUR 77/ wird das Gesamtnetz so in Teilnetze zerlegt, daß die Aktivitäten zwischen den Teilnetzen nahezu vernachlässigbar sind gegenüber den Aktivitäten innerhalb der Teilnetze. Solche Systeme nennt man fast-vollständig-zerlegbar. Bei der Analyse solcher Netze analysiert man die Teilnetze völlig isoliert vom Gesamtnetz. Man tut so, als ob es keine Aktivität zwischen dem Teilnetz und seiner Umgebung gäbe. Da diese Aktivitäten zwischen den Teilnetzen aber sehr gering sind, ergeben sich aus deren Vernachlässigung nur kleine Fehler. Diese Methode nennt man auch Aggregation.

Ein anderes Dekompositionsverfahren zur Analyse von Netzen mit nicht-exponentiell verteilten Bedienzeiten wird in /KÜHN 79/ vorgeschlagen. Die näherungsweise Bestimmung der Leistungsgrößen erfolgt bei diesem Verfahren durch getrennte Untersuchungen der einzelnen Stationen. Dies gelingt durch eine Approximation des Variationskoeffizienten des Ankunftsprozesses und des Abgangsprozesses. Hierzu wird der Abgangsprozess entsprechend den Übergängen zwischen den Stationen in Teilprozesse zerlegt und die Variationskoeffizienten dieser Teilprozesse bestimmt. Aus den Variationskoeffizienten der Teilprozesse der bei einer Station ankommenden Aufträge wird der Variationskoeffizient des Ankunftsprozesses ermittelt, der dann zusammen mit dem Variationskoeffizienten des Bedienprozesses die Bestimmung des Variationskoeffizienten des Abgangsprozesses ermöglicht. Zu Beginn dieser iterativen Vorgehensweise werden alle unbekannten Variationskoeffizienten der Ankunftsprozesse gleich eins gesetzt.

Da die MVA für eine große Anzahl von Aufträgen in Produktformnetzen sehr aufwendig ist, hat /SCHW 79/ eine Approximation der MVA für Netze mit lastunabhängigen Stationen vorgeschlagen. Bei dieser Vorgehensweise wird die Iteration nicht mit 0 wie bei der MVA, sondern mit einem bestimmten Initialwert gestartet. Dies bringt eine große Rechenzeitverkürzung und Speicherplatzbedarfsreduzierung.

Operationelle Analyse

Eine andere Methode zur Analyse von Warteschlangennetzen und zur Errechnung der Leistungsparameter ist die Operationelle Analyse, die von /BUZE 76/ eingeführt und von /DENN 78/ verfeinert wurde. Im Gegensatz zu den anderen Methoden werden hier keine statistischen Annahmen gemacht; vielmehr beobachtet man das existierende System während eines gewissen Zeitraumes und mißt sogenannte Basisgrößen. Aus diesen Größen werden dann die Leistungsgrößen ermittelt. Durch diese Vorgehensweise erhält man auch ohne wahrscheinlichkeitstheoretische Annahmen Produktformlösungen für Warteschlangennetze. Das Problem bei der Anwendung der operationellen Analyse bei existierenden Systemen ist, daß man einen geeigneten Beobachtungszeitraum finden muß, damit die errechneten Leistungsgrößen richtig sind.

Wenn das reale System aber nicht existiert, macht man Annahmen wie bei den stochastischen Methoden "Einzel-schrittverhalten" und "Homogenitätseigenschaften" und erhält damit die Leistungsgrößen. Damit bin ich der Meinung wie /HERZ 83/, daß diese Vorgehensweise eine einfache, didaktisch geschickte Beschreibung der Markov'schen Methoden für einen Personenkreis ohne mathematischen Hintergrund ist. Es gibt bei der Operationellen Analyse kein Ergebnis, das nicht zuvor mit den stochastischen Methoden hergeleitet wurde.

Asymptotische und "Job Bound" Analyse

Manchmal genügen Informationen über die Grenzen der erreichbaren Leistungsfähigkeit, um das System zu beurteilen. Eine Alter-

native zur exakten Lösung des Modells ist, Grenzen für den Leistungsumfang eines Modells zu erhalten. Eine solche "Begrenzungs-technik" ist die Asymptotische Analyse (ABA) /MUNT 74/. ABA benötigt als Eingaben die maximale relative Auslastung einer Einzelstation und die Summe der relativen Auslastungen aller Stationen. Als Ausgabe liefert sie die oberen Grenzen des Durchsatzes und die unteren Grenzen der mittleren Verweilzeit. Als einzige Voraussetzung, die erfüllt werden muß, um ABA anwenden zu können, müssen die relativen Auslastungen unabhängig von der Anzahl der sich im System befindenden Aufträge sein. Durch diese Voraussetzung werden die Ergebnisse ungenau und die Grenzen breit. /ZAHO 81/ haben ausgehend von der MVA die "Job Bound" Analyse vorgeschlagen, die eine schnelle Bestimmung eines engen Bereiches der möglichen Leistungsumfangswerte erlaubt. Mit Hilfe dieser Technik kann man eine obere und untere Grenze für den Durchsatz und für die mittlere Verweilzeit bestimmen.

Simultane Betriebsmittelbelegung

Man muß bei der Modellierung von Rechensystemen manchmal auch berücksichtigen, daß ein Auftrag gleichzeitig (simultan) zwei oder mehr Betriebsmittel belegen kann. Solche Warteschlangennetzmodelle können nicht mit den Produktformalgorithmen analysiert werden. /JACO 82/ haben eine approximative Lösungstechnik "Method of Surrogates" für Warteschlangennetze mit überlappender Bedienstationsbelegung entwickelt. Der Grundgedanke ist die Einteilung der Wartezeit in Primär- und Sekundärstationen, je nach Verantwortlichkeit der Bedienstationen. Die Primärstation ist die, die besetzt werden muß, bevor eine Bedienstation im Sekundärsystem benutzt werden kann.

Bei dieser Methode ist die Konvergenz allerdings nicht gesichert. Außerdem neigt sie zum Unterschätzen des Durchsatzes in der Größenordnung von 5-11%.

Netze mit Blockierungen

Die oben beschriebenen Methoden gehen immer davon aus, daß die Kapazitäten der Warteschlangen keinen Beschränkungen unterliegen. Wenn die Kapazitäten der Warteschlangen endlich sind, können nach einer gewissen Zeit Blockierungen vorkommen; d.h. wenn die Kapazität einer Warteschlange erschöpft ist, kann sie keinen neuen Auftrag mehr aufnehmen. Der Auftrag, der in diese Station kommen sollte, muß in der augenblicklichen Station verweilen und diese blockieren, bis in der vollen Station ein Platz frei wird. Solche Netze zu analysieren wird einer der Schwerpunkte dieser Arbeit sein. Der Fall, daß ein Auftrag wegen einer vollen Station verlorenggeht, wird in diesem Zusammenhang nicht berücksichtigt. Einige früheren Lösungsvorschläge werden zu Beginn des 4. Kapitels genauer beschrieben.

1.3 ÜBERSICHT ÜBER DIESE ARBEIT

In dieser Arbeit wird ein Multiprozessorsystem analytisch untersucht, in dem Prozesse sich über Puffer, sogenannte Kommunikationswarteschlangen, synchronisieren und aufgrund der endlichen Kapazitäten der Puffer Blockierungen vorkommen können, sowie es z.B. beim iAPX-432 Multiprozessorsystem der Fall ist.

Gemäß dem im Abschnitt 1.2 beschriebenen Modellierungsprozeß wird zunächst ein Systemmodell entwickelt. Dieses Systemmodell, d.h. das reale Multiprozessorsystem mit detaillierten Eigenschaften, wird in Kapitel 2 beschrieben.

Aus diesem Modell wird wieder entsprechend dem Modellierungsprozeß in Kapitel 3 ein Warteschlangennetzmodell entwickelt. Da in diesem Falle Prozessoren, Prozesse und Nachrichten im System vorkommen, kann das ganze Systemmodell nicht auf ein einziges Warteschlangennetzmodell abgebildet werden. Daher wird das System hierarchisch analysiert, d.h. es werden zwei Modelle (Gesamt-/Software Modelle) entwickelt. Zunächst wird das Software-Modell analysiert. Dann werden die Ergebnisse aus dem Software-Modell in das Gesamtmodell eingesetzt und die Leistungsgrößen für das Multiprozessorsystem berechnet.

Das Gesamtmodell besteht aus N Bedienstationen, wobei die erste Station die m identischen Prozessoren mit einer einzigen Warteschlange (als $M/M/m$ -PS) und die Stationen 2 bis N die wegen der Blockierung vorkommenden Verzögerungen (als $M/M/IS$ -.) darstellen. Dieses Modell hat unter vorgegebenen Prozessorbedienzeiten und zunächst angenommenen Bedienzeiten (=Blockierzeiten, Verzögerungszeiten) der Stationen 2 bis N und Übergangswahrscheinlichkeiten (=Blockierungs-, und Deblockierungswahrscheinlichkeiten) Produktformlösungen, da die beiden Stationstypen die Bedingungen des Produktformansatzes erfüllen.

Um die im Gesamtmodell zunächst angenommenen Werte für Blockierzeiten und Blockierwahrscheinlichkeiten zu erhalten, wird im Abschnitt 3.2 ein Software-Modell entwickelt, in dem Prozesse als abstrakte Bedieneinheiten, die Puffer als Warteschlange, und die Nachrichten als Aufträge modelliert werden. Damit erhalten wir ein geschlossenes Warteschlangennetz mit endlichen Kapazitäten. Für solche Netze gibt es weder Produktformlösungen noch eindeutige einfache Methoden. Einer der Schwerpunkte dieser Arbeit wird es sein, Algorithmen für die Analyse der Netze mit endlichen Warteschlangenkapazitäten zu entwickeln und daraus die erforderlichen Blockierwahrscheinlichkeiten und Blockierzeiten zu bestimmen. Dafür werden in Kapitel 4 drei Methoden präsentiert, von denen die erste (Zustandsraumtransformationmethode) exakte Lösungen für 2-Stationen-Netze und sehr gute approximative Werte (1-8% Abweichungen) für Mehrstationen-Fälle liefert.

Die zweite Methode MVABLO (Mittelwertanalyse für Blockiernetze) ist die Erweiterung der bekannten Mittelwertanalyse auf Blockiernetze. Diese Vorgehensweise liefert approximative Werte, die durchschnittlich um 1-20% von den simulativen Ergebnissen abweichen.

Die dritte Methode basiert auf der Produktform. Die Blockiernetze haben einen begrenzten Zustandsraum über den die Wahrscheinlichkeiten und daraus die Leistungsgrößen bestimmt werden können. Im Abschnitt 4.4 wird zunächst der Beweis für die Produktformlösung durchgeführt und danach drei effiziente Algorithmen entwickelt, mit denen dann die Normalisierungskonstante bestimmt werden kann. Neue Formeln werden für Leistungsgrößen

eingeführt, die die ermittelte Normalisierungskonstante benutzen. Die Ergebnisse werden mit der Simulation verglichen und es wird sich zeigen, daß sie Abweichungen von durchschnittlich 1-30% haben. Der Grund für schlechte Ergebnisse liegt darin, daß man die Abhängigkeiten zwischen den Stationen nicht berücksichtigt, was natürlich für Blockiernetze eine nicht realistische Annahme ist.

In Kapitel 5 wird das Gesamtmodell anhand 3 verschiedener numerischer Beispiele analysiert und mit Hilfe von Simulationsergebnissen validiert. Da das Gesamtmodell die Bedingungen für Produktformlösung erfüllt, können wir für die Analyse bekannte Algorithmen wie z.B. MVA oder auch die im Anhang A1 eingeführten Algorithmen anwenden. Die im Anhang A1 behandelten Algorithmen sind nicht nur auf die Analyse des Gesamtmodells beschränkt, sondern sie sind vielmehr auf eine breitere Klasse von Netzen anwendbar. Dabei werden bereits existierende Methoden erweitert, um den Aufwand zu reduzieren bzw. zusätzliche Stationstypen zu berücksichtigen. Im Anschluß an die Beschreibung jeder Methode wird eine Bewertung stattfinden. In der Bewertung werden die Rechenzeit, Speicherplatz, Gesamtheit der Ergebnisse angesprochen und die herkömmlichen Methoden zum Vergleich herangezogen.

2. BESCHREIBUNG DES MULTIPROZESSORSYSTEMS

Das zu analysierende Multiprozessorsystem soll den folgenden Aufbau besitzen:

- a) Das System besteht aus m identischen Prozessoren.
($m=2, \dots, 40$).
- b) Die Anzahl der im System zirkulierenden Prozesse ist konstant.
- c) Der gesamte zur Verfügung stehende Speicherplatz wird als unendlich groß angesehen. Dadurch entstehen keine Warteschlangen für den Eintritt in den Hauptspeicher.
- d) Die Prozesse können sich im System in 3 Zuständen befinden:
bereit, aktiv, blockiert.

Dadurch erhalten wir das folgende klassische Zustandsdiagramm für Prozesse:

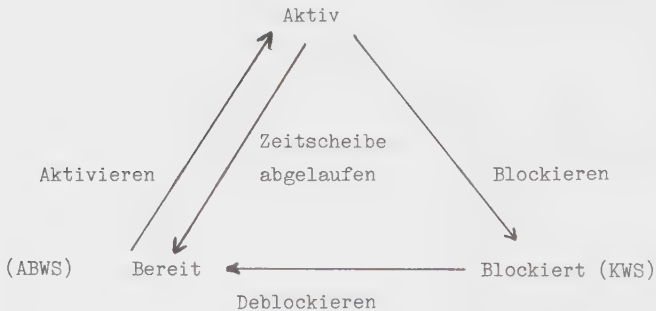


Fig. 2.1 Zustandsdiagramm der Prozesse

Im folgenden werden diese Prozeßzustände und Zustandsübergänge genauer erläutert.

Im System befinden sich 2 Arten von Warteschlangen:

1. Ablaufwarteschlange (Bereit-Zustand)

Sie verwaltet die physikalische Vergabe von Prozessoren. In der Ablaufwarteschlange befinden sich Prozesse, die auf Abarbeitung warten. Die Strategie der Ablaufwarteschlange ist PS. Prozesse sind eingereiht in Ablaufwarteschlange und werden den unterschiedlichen Prozessoren zur Ausführung zugeordnet. Ein Prozessor, der frei geworden ist, nimmt immer den ersten der eingereihten Prozesse. In der Fig. 2.1 stellt die Ablaufwarteschlange den Zustand bereit dar.

2. Kommunikationswarteschlangen (Blockiert-Zustand)

In Multiprozessorsystemen ist es üblich, daß die Prozesse, die sich im System befinden, miteinander in Verbindung treten können, um eine Synchronisation zu erreichen oder um irgendwelche Nachrichten (z.B. Rechenergebnisse) auszutauschen.

Im zu analysierenden System werden zur Prozeßkommunikation sogenannte Kommunikationswarteschlangen (KWS) benutzt. Jedem Prozeß wird eine solche Kommunikationswarteschlange zugeordnet, in die die anderen Prozesse in dem Zeitraum, in dem sie einen Prozessor belegen, d.h. aktiv sind, Botschaften (Nachrichten) ablegen können.

Dieses Ablegen entspricht einer SEND-Operation, die folgende Form hat:

SEND (Botschaft,Ziel)

Die Botschaften, die sich in einer KWS befinden, können nur von dem Besitzer der KWS entnommen werden, wenn er einen Prozessor belegt. Dieses Entnehmen einer Botschaft nennt man eine WAIT -Operation, die folgende Form hat:

WAIT (Botschaft)

Die Botschaften werden dabei nach FCFS abgearbeitet.

Die Kommunikationswarteschlangen haben endliche Kapazitäten, was bewirkt, daß sich die Prozesse bei der Durchführung der Synchronisationsoperation blockieren können. Dabei verlassen sie die Prozessoren und gehen in den Blockierzustand über. (Fig. 2.1)

Zustandsübergänge

Es gibt 2 Arten von Blockierungen bzw. Deblockierungen im System, die in der folgenden Definition ersichtlich sind:

Definition

Der Prozeß i (für $i=1, \dots, K$) kann unter folgenden Bedingungen seine Zustände ändern:



Wie in dieser Definition ersichtlich laufen beide Synchronisationsoperationen unter gegenseitigem Ausschuß.

Informell

- i) Da die Kommunikationswarteschlangen nur eine begrenzte Kapazität besitzen, kann es sein, daß eine SEND-Operation an eine KWS-j gerichtet ist, die zu diesem Zeitpunkt keinen Platz mehr für die Aufnahme einer Botschaft hat.

In diesem Fall gibt der Prozeß-i, der die SEND-Operation ausführen wollte, den Prozessor auf, hängt sich an die KWS-j und wartet, bis in der KWS-j ein Platz für seine Botschaft frei wird. Wird der Platz verfügbar, wird der Prozeß-i deblockiert d.h. er geht in den Bereitzustand (ABWS) über.

- ii) Ebenso kann es vorkommen, daß ein Prozeß eine WAIT-Operation durchführen will und seine KWS keine Botschaft enthält. In diesem Fall verläßt er den Prozessor, geht in seine eigene KWS und wartet bis eine Botschaft an ihn gesendet wird. Wenn eine Botschaft ankommt, deblockiert sich der Prozeß-i und geht in den Bereit-Zustand (ABWS) über.

In dieser Arbeit wird das oben beschriebene Multiprozessorsystem analytisch untersucht und Leistungsgrößen wie z.B.

Auslastung der Prozessoren
Durchsatz des Systems
Mittlere Verweilzeiten der Prozesse
Mittlere Wartezeiten der Prozesse
Mittlere Warteschlangenlängen
Blockierzeiten und Blockier-
wahrscheinlichkeiten der Prozesse

ermittelt. Das Gesamtsystem wird simuliert und die analytisch erhaltenen Leistungsgrößen validiert.

3. MODELLIERUNG DES MULTIPROZESSORSYSTEMS

Aus dem Systemmodell für das Multiprozessorsystem, Kap.2, muß gemäß dem Modellierungsprozeß, Abschnitt 1.3, ein Warteschlangenmodell entwickelt werden. Der Prozeßablauf durch das System kann wie folgt modelliert werden: (Fig.3.1)

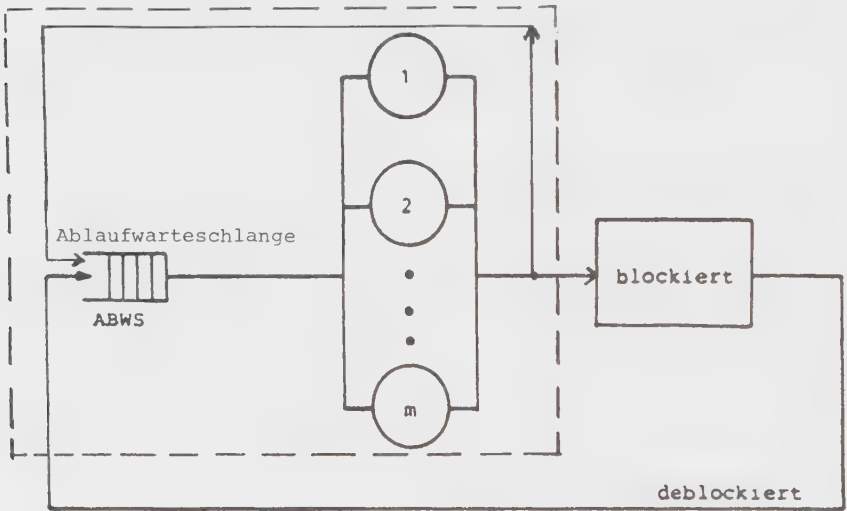


Fig. 3.1 Warteschlangenmodell für den Prozeßablauf

Die Ablaufwarteschlange repräsentiert den Zustand bereit, die Prozessoren den Zustand aktiv und Verzögerungen aufgrund der Synchronisation den Blockierzustand.

Nach dem Ablauf der Zeitscheibe wird ein Prozeß zurück in die Ablaufwarteschlange eingereiht. Bei Nichtausführbarkeit einer anstehenden synchronisierenden Aktion (Diese sind SEND bzw. WAIT von Nachrichten) wird ein Prozeß die Prozessoren verlassen und Blockierzustand betreten, d.h. sich blockieren. Nach einer bestimmten Verweilzeit (=Blockierzeit) im Blockiertzustand werden die Prozesse wieder zur weiteren Bearbeitung zur Ablaufwarteschlange zurückkehren.

Da es nicht möglich ist, einerseits den Prozeßablauf und

Es fällt auf, daß hier die Prozesse nicht mehr miteinander kommunizieren, sondern ein relativ einfaches Warteschlangennetz durchlaufen. Somit müssen die Einflüsse der Prozeßkommunikation auf die Leistungsgrößen des Gesamtmodells implizit durch eine geeignete Wahl der Übergangswahrscheinlichkeiten sowie der Bedienzeiten der Verzögerungsstationen berücksichtigt werden.

Zwei Wege eines Prozesses nach Verlassen der Prozessoren sind möglich: entweder kehrt der Prozeß direkt in die Ablaufwarteschlange zurück oder er betritt die Kommunikationswarteschlangen, wo er blockiert wird.

Die Verzweigungswahrscheinlichkeit p_{11} , daß der Prozeß nicht blockiert wurde, kann aus der folgenden Formel berechnet werden:

$$p_{11} = 1 - \frac{\sum_{i=2}^N P_{B_i}}{N-1} \quad (3.1)$$

wobei P_{B_i} die Wahrscheinlichkeit ist, daß sich der i -te Prozeß bei seiner Bearbeitung blockiert.

Bei der Analyse des Gesamtmodells (Fig. 3.2) sind die einzelnen Prozesse nicht mehr identifizierbar, lediglich ihre Gesamtzahl stimmt mit der Zahl der an der Kommunikation beteiligten Prozesse überein. Somit können die Prozesse nicht mehr direkt ihren jeweiligen Verzögerungsstationen zugewiesen werden, sondern nur noch durch die Angabe von Übergangswahrscheinlichkeiten. Damit ein Prozeß mit hoher Blockierwahrscheinlichkeit auch relativ häufig seine eigene Verzögerungsstation betritt, sind die Wahrscheinlichkeiten des Betretens einer solchen Station mit den Blockierwahrscheinlichkeiten der einzelnen Prozesse zu gewichten:

$$p_{1i} = \underbrace{\frac{\sum_{i=2}^N P_{B_i}}{N-1}}_a \cdot \underbrace{\frac{P_{B_i}}{\sum_{i=2}^N P_{B_i}}}_{b} = \frac{P_{B_i}}{N-1} \quad \text{für } i=2, \dots, N \quad (3.2)$$

- wobei a die Wahrscheinlichkeit dafür angibt, daß überhaupt eine Blockierung auftritt.
- b die Wahrscheinlichkeit wiedergibt, daß bei Blockierung an Station i verzögert wird.

Für die Analyse des Gesamtmodells, Fig. 3.2 werden im Anhang A1 neue Lösungsalgorithmen entwickelt bzw. Nachteile bereits existierender Algorithmen beseitigt. Die Algorithmen sind nicht nur auf dieses Gesamtmodell, sondern auf eine viel breitere Klasse von Warteschlangennetzen anwendbar.

Die Hauptschwierigkeit liegt in folgender Frage:
Wie können die Blockierwahrscheinlichkeiten und Blockierzeiten der einzelnen Prozesse im Gesamtmodell ermittelt werden?

Für diesen Zweck wurde ein Software-Modell entwickelt, das die Prozeßkommunikation darstellt.

3.2 SOFTWARE-MODELL (MODELL FÜR DIE PROZEßKOMMUNIKATION)

Um die Blockierzeiten und Blockierwahrscheinlichkeiten der Prozesse zu berechnen, wird das Modell (Fig. 3.3) für die reine Prozesskommunikation entwickelt.

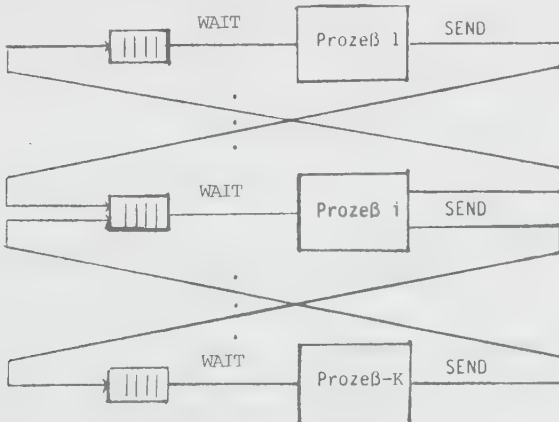


Fig.3.3 Modell für die Prozeßkommunikation

Dabei werden die folgenden Annahmen gemacht:

- Die Prozesse sind abstrakte Bedieneinheiten ($m_i=1$). Es gibt K kommunizierende Prozesse. Jeder Prozeß hat eigene Kommunikationswarteschlange, die eine endliche Kapazität M_i hat. Die Warteschlangenstrategie ist FCFS. Wenn die Zahl M_i in der KWS-i erreicht ist, blockiert sich ein Prozeß, der gerade an den Prozeß-i eine Nachricht senden möchte.
- Die Nachrichten sind Aufträge, die durch das System zirkulieren. Die Anzahl der Nachrichten ist konstant,

$$\sum_{i=1}^K c_i = C$$

- Die mittlere Bedienzeit der jeweiligen Bedienstation ist die mittlere Zeit zwischen dem WAIT und SEND von Nachrichten, die exponentiell verteilt ist mit der Bedienrate μ_i (für $i=1, \dots, K$)
- Ein Prozeß i sendet eine Nachricht an den Prozeß j mit der Wahrscheinlichkeit $p_{ij} = 1/(K-1)$ (für $i, j=1, \dots, K$); ($p_{ij} = 0$, wenn $i=j$).
- Der Prozeß i kann sich blockieren, wenn er an den Prozeß j eine Nachricht senden möchte und die KWS-j voll ist bzw. wenn er eine Nachricht aus eigener Kommunikationswarteschlange empfangen möchte und die Warteschlange leer ist.
- Der Systemzustand ist definiert durch

$$c = (c_1, c_2, \dots, c_K)$$

wobei c_i ($i=1, \dots, K$) die Anzahl der Nachrichten in der i-ten Bedienstation ist.

- Ein Zustand c ist "durchführbar", wenn

$$\sum_{i=1}^K c_i = C \wedge (c_i < M_i \vee c_i > 0) \quad \text{für } i=1, 2, \dots, K$$

Wie aus der Modellbeschreibung ersichtlich, haben wir als Software-Modell ein Warteschlangennetz mit Stationen endlicher Kapazität erhalten. Bekanntlich gibt es in der Literatur keine eindeutige, einfache Lösungen für solche Netze. Einer der Schwerpunkte dieser Arbeit wird es unter anderem sein, Algorithmen für solche Netze zu entwickeln. In Kapitel 4 werden 3 Methoden eingeführt, von denen eine unter bestimmten Bedingungen exakte Lösungen liefert.

SPEZIALFALL

Es wird auch der Fall der Prozeßkommunikation analysiert, wo jeweils nur 2 Prozesse miteinander kommunizieren. Dadurch erhält man mehrere 2-Stationen-Warteschlangennetze als Software-Modell, wie in der folgenden Fig. 3.4 dargestellt ist.

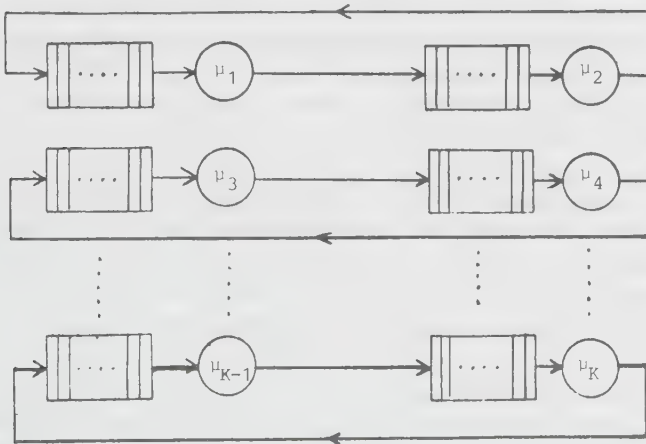


Fig. 3.4 Software-Modell mit Kommunikationen
jeweils von 2 Prozessen

4. ALGORITHMEN FÜR DIE ANALYSE DES SOFTWARE MODELLS

Um die Blockierzeiten und Blockierwahrscheinlichkeiten der einzelnen Prozesse zu berechnen, wurde in Abschnitt 3.2. ein Software-Modell entwickelt. Dabei wurde ein geschlossenes Warteschlangennetz mit endlichen Warteschlangenkapazitäten geschaffen.

Obwohl bisher viele Arbeiten veröffentlicht wurden, gibt es keine allgemeingültigen, einfachen Algorithmen für geschlossene Netze mit endlichen Warteschlangenkapazitäten. Im folgenden möchte ich eine kurze Übersicht über die wichtigsten bisherigen Arbeiten geben.

In /KONH 76,77/ wird ein Warteschlangennetz mit einem unendlichen Puffer vor einer zyklischen Anordnung von zwei Bedienstationen untersucht. Die Analyse wird unter der Annahme exponentieller Verteilungen durchgeführt. Wenn die Anzahl der Aufträge in den zwei Bedienstationen einen Maximalwert M erreicht, werden neu ankommende Aufträge im Puffer zurückgehalten, d.h. blockiert. Ein Algorithmus zur Berechnung der stationären Zustandswahrscheinlichkeiten wird angegeben und einige Spezialfälle erarbeitet. Die Vorgehensweise liefert aber nur für die betrachteten Spezialfälle exakte Ergebnisse. Außerdem ist der Algorithmus komplex und sehr aufwendig.

/PITT 79/ betrachtet zwei geschlossene Markoff'sche Warteschlangennetzmodelle mit Blockierung. Im ersten Modell geht ein Auftrag in die gleiche Station wieder zurück, wenn die Station, in die er eintreten möchte, voll besetzt ist. Im zweiten Modell geht ein Auftrag zu einer beliebigen anderen Station, wenn die Station, zu der er übertreten wollte, schon besetzt ist. Die Zeit, die ein Auftrag für das Suchen einer leeren Station bringt, wird in diesem Modell vernachlässigt. /PITT 79/ erhält eine asymptotische Lösung für die Blockierwahrscheinlichkeiten durch Anwendung der nichtlinearen mathematischen Programmiermethoden.

/PERR 81/ untersucht ein offenes Markoff-Netzwerk, das aus N symmetrischen, parallelen Bedienstationen als erste Ebene und einer Station als zweiter Ebene besteht. Die Blockierung der Aufträge in der ersten Ebene kommt jedesmal dann vor, wenn ein Auftrag in der ersten Ebene fertig bedient worden ist. Die Sta-

tion bleibt solange blockiert, bis der Auftrag auch in der zweiten Ebene fertig bedient worden ist. /PERR 81/ erhält eine Approximationsmethode für die Wahrscheinlichkeitsverteilung der Anzahl der blockierten Bedienstationen in der ersten Ebene.

/NILS 81/ behandelt das Problem der Blockierung in offenen Warteschlangennetzen mit exponentiellen Bedienzeitverteilungen. Die Arbeit basiert auf Kommunikationsnetzen, wobei jede Station seriell geschaltet ist. Er benutzt eine COX-Verteilung für jede Station, wobei die zweite Phase die Blockierung der entsprechenden Station modelliert. Die Blockierwahrscheinlichkeiten ermittelt man durch eine iterative Formel und die Bedienzeiten der zweiten Phasen sind gleichgesetzt mit den Bedienzeiten der ersten Phasen. Für die Analyse setzt er die Methode von Marie /MARI 80/ ein. Die Ergebnisse weichen manchmal um 10 - 20% von den Simulationswerten ab.

/TAKA 80/ entwickelt eine einfache Methode für die approximative Analyse der offenen Netzwerke mit Blockierung. Sei P_{Bj} die Wahrscheinlichkeit, daß die Station j voll ist. Jede Station wird als eine M/M/1-Warteschlange mit endlicher Kapazität behandelt, deren Ankunftsrate und mittleren Bedienzeiten in Termen der Wahrscheinlichkeiten $P_{B1}, P_{B2}, \dots, P_{Bj}$ ausgedrückt werden. Diese Wahrscheinlichkeiten werden wiederum in Termen der Ankunftsrate und mittleren Bedienzeiten ausgedrückt. Dies führt zu einem Gleichungssystem mit J nichtlinearen Gleichungen, dessen Lösung eine Approximation für die $P_{B1}, P_{B2}, \dots, P_{Bj}$ liefert. Approximative Werte für andere Leistungsgrößen erhält man aus diesen Wahrscheinlichkeiten. Die Ergebnisse weichen auch in diesem Fall von den Simulationswerten ab.

/BOXM 81/ führt eine andere Vorgehensweise für die Behandlung der offenen Netze mit Blockierung an. Wenn bei einer Station j die Kapazität erreicht ist, werden alle Stationen blockiert, die Aufträge bedienen, die zur Station j übergehen werden. Außerdem werden die externen Ankünfte zu einer vollen Station ignoriert. Es werden zwei Phasen für eine Station eingeführt, in denen sich das Gesamtsystem abwechselnd befindet. Zunächst werden die Wahrscheinlichkeiten geschätzt, daß sich das System in Phase 2 befindet und dann die Wahrscheinlichkeit approximiert, mit der sich höchstens j Aufträge im System befinden. Da die Vorgehensweise

äußerst kompliziert ist, möchte ich die 2 Phasen noch näher erläutern:

Die Menge N der Stationen wird in 2 Untermengen N' und N'' aufgeteilt. (N', N'') tritt in Phase 2 ein, wenn eine der Stationen aus N'' gesättigt ist oder ein Auftrag, der als nächstes für eine Station in N'' bestimmt ist, im Augenblick kurz vor oder schon in der Bedienung in einer Station aus N' ist. (N', N'') geht in die Phase 1 über, wenn alle Stationen aus N' , die mit Stationen aus N'' verbunden sind, leer sind. Die Methode funktioniert aber nur unter der Bedingung, daß ein Auftrag nicht mehr wieder in die gleiche Station zurückkommt. Die Genauigkeit der Methode liegt für 2 bis 3 Stationen ungefähr bei 10 %. Für größere Netze gibt es große Fehlerabweichungen.

Die Isolationsmethode /LABE 80/ und ihre Erweiterung /GEOR 81/ ist geeignet, Paketvermittlungsnetze mit Blockierung für Kommunikationssysteme zu analysieren. Die Gesamtvorgehensweise basiert darauf, daß das System in einzelne Stationen zerlegt wird. Die Parameter der isolierten Station werden so angepaßt, daß der Einfluss der übrigen Stationen des Netzes mitberücksichtigt wird. Die so entstandene G/G/1/m-Station wird mit Hilfe der Diffusionsapproximation analysiert. Ein Auftrag, der von Station i fertig bedient worden ist und nach Station j übergehen möchte, dieses aber nicht kann, da Station j bereits voll besetzt ist, wird von Station i ein weiteres Mal bedient. Die Wahrscheinlichkeiten der Rückweisung eines Auftrages ist beim ersten Versuch gleich der Wahrscheinlichkeit, daß die Warteschlange der Station j voll besetzt ist ($P_{\text{Full}j}$). Bei den weiteren Versuchen wird die Information, daß die Station j eine Bedienzeit zuvor noch voll war, mit in die Rückweisungswahrscheinlichkeit einbezogen. Die Ergebnisse weichen von den Simulationswerten um 10 - 25 % ab.

/HORD 81/ beschäftigt sich mit geschlossenen Netzwerken aus Stationen mit exponentieller Bedienzeitverteilung mit Blockierung. Die Blockierung der Aufträge an einer Station soll nur von der Zahl der Aufträge in dieser Station abhängen. Für diese Netze stellen sie das globale Gleichgewichtsgleichungssystem auf und lösen es nach Produktformeln, wobei die Gesamtvorgehensweise auf den möglicher ausführbaren Zuständen basiert. In der Arbeit werden nur Produktformllösungen gegeben und keine explizite Angaben,

wie die Normalisierungskonstante und die Leistungsgrößen bestimmt werden sollen. In Abschnitt 4.4. werde ich - basierend auf der Lösung von /HORD 81/ - drei Algorithmen für die Berechnung der Normalisierungskonstanten und neue Formeln für die Leistungsgrößen einführen.

Es gibt noch weitere Arbeiten, die sich mit Blockiernetzen befassen. Leider kann ich hier nicht alle ausführlich beschreiben. Mit der obigen detaillierten Beschreibung der Methoden wollte ich zeigen, daß die existierenden Lösungen irgendwelche Nachteile oder Beschränkungen haben und sich vorwiegend auf offene Netze beziehen.

Im Softwaremodell (Abschnitt 3.2.) hatte ich die Prozesse als abstrakte Bedieneinheiten angesehen, die Puffer als Kommunikationswarteschlangen und die Nachrichten als Aufträge nachgebildet. Ferner hatte ich angenommen, daß jeder Prozeß eine eigene Warteschlange besitzt, die die Nachrichten nach der Strategie FCFS verarbeitet. Die wichtigste Annahme ist die, daß die Kapazitäten der Warteschlangen endlich sind. Damit hatten wir ein geschlossenes Netz mit endlichen Warteschlangenkapazitäten, d.h. ein Warteschlangennetz mit Blockierung erhalten.

Da die bisher entwickelten Methoden zur Analyse solcher Netze für meine Zwecke unzureichend sind, kann ich bei der Auswahl einer Methode nicht auf die Literatur zurückgreifen. Daher werde ich in diesem Abschnitt neue Methoden vorschlagen und die ermittelten Ergebnisse mit der Simulation validieren.

Folgende Annahmen gelten im gesamten Kapitel 4:

- Das System besteht aus N Bedienstationen.
- Die Anzahl der Aufträge im System ist konstant und beträgt K .
- Jede Station hat eine Bedieneinheit ($m_i=1$) und exponentielle Bedienzeitverteilung mit Mittelwert $1/\mu_i$ ($i=1, \dots, N$).
- Jede Station hat die endliche, konstante Kapazität M_i ($i=1, \dots, N$).
- Ein in der Bedienstation i fertig bedienter Auftrag geht mit der Wahrscheinlichkeit p_{ij} zur Station j über, falls in der j -ten Station die Anzahl der Aufträge die Kapazität M_j nicht erreicht ist. Sonst blockiert sich der Auftrag in der i -ten

Station solange, bis in der j-ten Station ein Auftrag fertig bedient und ein Platz für diesen Auftrag verfügbar wird. Falls sich mehrere Aufträge an ein und derselben Station blockieren, so werden sie nach FCFS deblockiert.

- Die Gesamtanzahl der Aufträge K im System muß kleiner sein als die Summe der Stationskapazitäten, d.h.:

$$K < \sum_{i=1}^N M_i$$

- Die Aufträge werden in den Stationen nach FCFS abgearbeitet.

4.1. DIE ZUSTANDSRAUMTRANSFORMATIONSMETHODE FÜR GESCHLOSSENE TANDEM- NETZE MIT BLOCKIERUNG

Wir betrachten ein geschlossenes Tandem-Netz mit 2 seriell geschalteten Bedienstationen. Alle Aufträge, die die erste Bedienstation verlassen, werden der zweiten zugeführt und in deren Warteschlange eingereiht und umgekehrt. Es können weder neue Aufträge von außen ins System gelangen noch aus diesem abfließen. Ansonsten gelten alle Annahmen, die zu Beginn des Kapitels 4 gemacht wurden.

Bekanntlich gilt für geschlossene Netze ohne Kapazitätsbegrenzung die folgende Binomialkoeffizientenformel, die besagt, wie viele Möglichkeiten es gibt, K Aufträge auf N Stationen zu verteilen:

$$\binom{N+K-1}{N-1} \quad (4.1)$$

Dieser Koeffizient liefert uns die Anzahl $|Z|$ der Zustände im System. Im Falle der Tandem-Netze mit 2 Stationen erhalten wir:

$$|Z| = (K + 1) \quad \text{Zustände.} \quad (4.2)$$

Die $(K+1)$ Zustände mit den Übergangsraten μ_1 bzw. μ_2 können wir wie in der folgenden Figur 4.1. zeichnen:

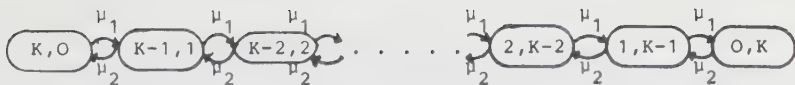


Fig. 4.1

wobei (i, j) den Zustand mit i Aufträgen in Station 1 und j Aufträgen in Station 2 bezeichnet.

Dem Diagramm ist zu entnehmen, daß vor jeder Bedienstation mindestens $(K-1)$ Warteplätze vorhanden sein müssen. Ein Auftrag befindet sich ja in der Bedieneinheit zur Bearbeitung.

Im realen Leben ist es üblich, daß alles endlich ist. So können wir auch hier nicht immer annehmen, daß die Stationen unendliche Kapazitäten haben, d.h. jede Station muß nicht in der Lage sein, alle Aufträge im System aufzunehmen. Eine natürliche Begrenzung der Warteschlangenkapazität ergibt sich durch den zur Verfügung stehenden Speicherplatz. Die Aufnahmekapazität der Station i sei daher M_i . M_i ist also die maximal mögliche Anzahl der Aufträge in der Warteschlange + der in Station i in Bedienung befindliche Auftrag ($i=1,2$).

Da insgesamt K Aufträge im System Platz finden müssen, gilt:

$$M_1 + M_2 > K \quad (4.3)$$

Die Anzahl k_i der zu einem Zeitpunkt in der Bedienstation i befindlichen Aufträge ist:

$$\max \{0, K - M_{(i+1 \bmod 2)}\} \leq k_i \leq \min \{M_i, K\}$$

Ferner gelten folgende Vereinbarungen:

$$\begin{aligned} M_1 &> 0 & M_2 &> 0 \\ K &> M_1 & K &> M_2 \end{aligned} \quad (4.4)$$

Unter diesen Voraussetzungen wird weder Bedienstation 1 noch Bedienstation 2 leer. Die Anzahl k_i der Aufträge, die sich momentan in Bedienstation i befinden, schwankt zwischen zwei Werten:

$$K - M_{(i+1 \bmod 2)} \leq k_i \leq M_i \quad (i=1,2) \quad (4.5)$$

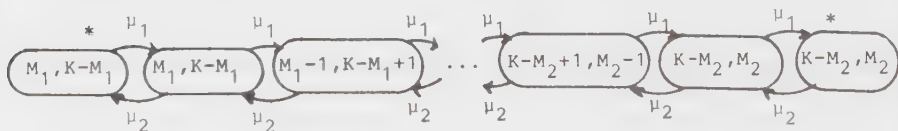
Wenn wir z.B. in einem Netz die Kapazitäten $M_1=4$ und $M_2=7$ und die Gesamtzahl der Aufträge $K=8$ wählen, haben wir den Fall

$$\begin{aligned} 1 &\leq k_1 \leq 4 \\ 4 &\leq k_2 \leq 7 \end{aligned}$$

Da wir Kapazitätsbeschränkungen haben, können wir nicht annehmen, daß alle Zustände ausführbar sind. Die ausführbaren Zustände des Blockiernetzes erhalten wir, indem wir im Zustandsdiagramm des entsprechenden Netzes ohne Blockierung (Fig.4.1.) nur die im Blockiernetz ausführbaren Zustände betrachten. Der Zustandsraum des Blockiernetzes ist also ein Teilgraph des Zustandsraumes desselben Netzes ohne Kapazitätsbeschränkung.

Man muss lediglich die Übergänge von Randzuständen (das sind Zustände, die die max. Kapazitäten M_1 bzw. M_2 beinhalten) zu den Nachbarzuständen (M_1+1) bzw. (M_2+1) berücksichtigen, die die Blockierungen verursachen. Das bedeutet, daß die Randzustände einen Zyklus machen und in sich selbst übergehen.

Anhand dieser Überlegungen läßt sich das komplette Zustandsdiagramm ableiten:



Figur 4.2

Es ist leicht ersichtlich, daß Fig.4.2 tatsächlich in Fig.4.1 enthalten ist. Wir setzen alle Zustände, die über den Kapazitäten liegen auf 0 bis auf die Blockierzustände (in Fig.4.3 mit * gekennzeichnet), die die unmittelbaren Nachbarn von ausführbaren Zuständen ohne Blockierung sind.

Im numerischen Beispiel bekommen wir folgenden Zustandsraum:

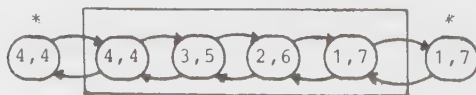


Fig. 4.3

Die mit * gekennzeichneten Zustände (4,4) und (1,7) bedeuten, daß die entsprechende Bedienstation 1 bzw. 2 sich blockiert.

Man muß darauf achten, daß ein gleichzeitiges Blockieren beider Bedienstationen nicht möglich ist, weil sonst ein Deadlock entstehen würde (unsere Netze sollen deadlockfrei sein).

Mit den bisherigen Überlegungen können wir das folgende Theorem aufstellen:

THEOREM

Zu einem gegebenen geschlossenen Tandem-Netz mit endlichen Warteschlangenkapazitäten existiert ein äquivalentes geschlossenes Netz ohne Blockierungen.

Beweis

Wir hatten bei unseren Überlegungen den Zustandsraum eines Netzes mit Blockierungen auf einen Teilzustandsraum (Figur 4.3.) reduziert.

Aus diesem Teilzustandsraum wissen wir die Anzahl der Zustände $|Z|$, die sich aus der Summe der Anzahl der möglichen Zustände plus der Anzahl der Blockierzustände ergibt.

Formal

$$\begin{aligned} |Z| &= 1 + (M_1 - (K - M_2)) + 2 \\ &= M_1 + M_2 - K + 3 \end{aligned}$$

Andererseits gilt in Netzen mit 2 Stationen Gl.(4.2).

Aus beiden Gleichungen erhalten wir die Gesamtanzahl der Aufträge $\hat{K}$ im äquivalenten geschlossenen Netz ohne Blockierung:

$$\hat{K} = M_1 + M_2 - K + 2 \quad (4.6)$$

Wenn wir den Zustandsraum für $\hat{K}$ zeichnen, sehen wir sofort, daß die Zustände genau den Zuständen im Blockiernetz (Figur 4.2) entsprechen.



Figur 4.4

Die Anzahl der Zustände und die Übergänge zwischen den Zuständen stimmen überein. Die Zustände, die eine leere Bedienstation repräsentieren ($(\hat{K}, 0)$ bzw. $(0, \hat{K})$) entsprechen den Blockierzuständen.

Systemgrößen, die nur von den Zustandswahrscheinlichkeiten und damit von den Übergangsraten der Bedienstationen abhängen, können direkt über ein äquivalentes Netz aus dem transformierten Zustandsraum ohne Blockierung berechnet werden. Zu diesen Systemgrößen zählen z.B. der Durchsatz λ und die Auslastung ρ . Um den Durchsatz $\lambda(\hat{K})$ zu bestimmen, können wir den Algorithmus TUPAL (Gl.(A1.3)) benutzen. Der Durchsatz $\lambda(\hat{K})$ aus dem Netz ohne Blockierung entspricht genau dem Durchsatz $\lambda(K)$ des Netzes mit Blockierung.

Die Zustandswahrscheinlichkeiten im Netz ohne Blockierung ergeben sich aus der folgenden Produktform:

$$p(k_1, k_2)^{\wedge} = \frac{1}{G(K)} \prod_{i=1}^N x_i^{k_i} \quad (4.7)$$

wobei x_i die Lösung der Gl.(A1.4) ist.

$\wedge$ bedeutet, daß die Zustandswahrscheinlichkeiten im Netz mit unendlichen Kapazitäten berechnet werden. Die errechneten Zustandswahrscheinlichkeiten $p(k_1, k_2)^{\wedge}$ im Netz ohne Blockierung entsprechen genau den Zustandswahrscheinlichkeiten $p(k_1, k_2)$ im Netz mit Blockierung. q.e.d.

Mit Hilfe der ermittelten Zustandswahrscheinlichkeiten können die Leistungsgrößen für Blockiernetze ermittelt werden.

LEISTUNGSGRÖSSEN FÜR BLOCKIERNETZE

Die mittlere Anzahl von Aufträgen ergibt sich aus

$$\bar{k}_i = \sum_{l=K-M_j}^{M_i} 1 \cdot p_i(l) \quad \text{für } i, j=1, 2 \quad i \neq j \quad (4.8)$$

wobei $p_i(l)$ die Randwahrscheinlichkeiten sind, daß sich in der i -ten Station l Aufträge befinden. Sie werden im Netz ohne Blockierung berechnet und durch Rücktransformation auf das Netz mit Blockierung abgebildet.

Unter Benutzung der Äquivalenz von Nichtblockier- und Blockiersystemen läßt sich Gl.(4.8) wie folgt umformen:

$$\begin{aligned} \bar{k}_i &= (K-M_j) P_{Bi} + \sum_{l=K-M_j}^{M_i} 1 p(l, K-1) + M_i P_{Bj} \\ &= (K-M_j) \frac{1}{G(\hat{K})} x_i^0 x_j^{M_1+M_2-K+2} \\ &\quad + \sum_{l=K-M_j}^{M_i} \frac{1}{G(\hat{K})} 1 x_i^{1-K+M_j+1} x_j^{M_j+1-1} \\ &\quad + M_i \frac{1}{G(\hat{K})} x_i^{M_1+M_2-K+2} x_j^0 \end{aligned}$$

wobei P_{Bi} die Blockierwahrscheinlichkeit der i -ten Station ist.
Durch Einsetzen der Gl.(4.7) erhalten wir dann:

$$\bar{k}_i = \frac{1}{G(\hat{K})} \left[(K-M_j) x_j^{\hat{K}} + M_i x_i^{\hat{K}} + \sum_{l=1}^{K-1} (K-M_j+1-l) x_i^l x_j^{\hat{K}-l} \right] \quad (4.9)$$

für $i, j = 1, 2 \quad i \neq j$

wobei $\hat{K}$ die Gesamtzahl der Aufträge im Netz ohne Blockierung ist (ergibt sich aus Gl.(4.6)).

$G(\hat{K})$ ist die Normalisierungskonstante, die sich aus den ermittelten Durchsatzgrößen $\lambda(j)$ mit $j = 1, \dots, \hat{K}$ wie folgt berechnen läßt:

$$G(\hat{K}) = \frac{1}{\sum_{j=1}^{\hat{K}} \lambda(j)} \quad (4.10)$$

Die im folgenden definierten Leistungsgrößen gelten für alle Blockiernetze, d.h. sind nicht beschränkt auf 2-Stationen-Fälle.

- Die mittlere Verweilzeit eines Auftrags in der i-ten Station erhält man aus Little's Gesetz:

$$\bar{t}_i(k) = \frac{\bar{k}_i(k)}{\lambda_i(k)} \quad \text{für } i=1,2 \quad (4.11)$$

- Die Auslastungen der Stationen ergeben sich aus:

$$\rho_i = \frac{\lambda_i}{\mu_i} \quad (4.12)$$

Da jede Bedienstation mindestens einen Auftrag enthält, läßt sich die Auslastung als Leerlaufanteil definieren. Es gelte deshalb für die Auslastung:

$$\rho_i = \frac{\text{Gesamtzeit} - \sum_{i=1}^N \text{Blockierzeit von Bedienstation } i}{\text{Gesamtzeit}} \quad (4.13)$$

- Damit läßt sich die Blockierwahrscheinlichkeit p_B der Bedienstation i ($i=1,2$) einfach berechnen:

$$p_{B_i} = 1 - \rho_i \quad (4.14)$$

- Die mittleren Blockierzeiten der i -ten Station können wie folgt ermittelt werden:

$$\bar{t}_B = \bar{t}_i^* - \bar{t}_i \quad (4.15)$$

wobei $\bar{t}_i^*$ die mittlere Verweilzeit der i -ten Bedienstation, die im gegebenen Netz ohne Berücksichtigung der Kapazitäten berechnet werden,

$\bar{t}_i$ die mittlere Verweilzeit der i -ten Station, die im Netz mit Berücksichtigung der Kapazitäten berechnet werden.

Die ganze Vorgehensweise können wir wie folgt zusammenfassen:

Gegeben sei ein geschlossenes 2-Stationen-Netz mit endlichen Kapazitäten M_1 bzw. M_2 und K Aufträgen.

- Berechne nach Gl.(4.6)

$$\hat{K} = M_1 + M_2 - K + 2$$

als Anzahl der Aufträge in einem äquivalenten 2-Stationen-Netz ohne Blockierung.

- Führe den TUPAL-Algorithmus für das Netz ohne Blockierung mit $\hat{K}$ Aufträgen aus und ermittle die Durchsatzwerte $\lambda(k)$ für $k = 1, \dots, \hat{K}$, wobei $\lambda(\hat{K})$ genau dem Durchsatz des Netzes mit beschränkten Kapazitäten $\lambda(K)$ entspricht.

- Bestimme mit Hilfe der $\lambda(\hat{k})$ -Werte die Auslastung $\rho_i(\hat{k})$ (mit Gl.(4.12)) und die Normalisierungskonstante $G(\hat{k})$ (mit Gl.4.10)). Diese Größen $\rho_i(\hat{k})$ und $G(\hat{k})$ gelten auch für das Blockiernetz.
- Berechne die mittlere Anzahl von Aufträgen mit Gl.(4.9), die mittlere Verweilzeit (Gl.(4.11)), die Blockierwahrscheinlichkeiten (Gl.(4.14)) und die Blockierzeiten (Gl.(4.15)).

Zur Verdeutlichung der bisherigen Definitionen hilft das folgende numerische Beispiel:

Beispiel

Gegeben sei ein geschlossenes Netz mit $N=2$ Stationen und $K=10$ Aufträgen. Die Bedienstationen haben eine Bedieneinheit ($m_i=1$) und exponentielle Bedienzeitverteilungen mit den Mittelwerten $1/\mu_1 = 2$ bzw. $1/\mu_2 = 0.9$ und FCFS-Strategien. Die erste Bedienstation hat die konstante Kapazität $M_1=7$ und die Bedienstation 2 $M_2=8$.

Nach Gl.(4.2) erhalten wir die folgende Anzahl der Zustände in diesem gegebenen Netz ohne Berücksichtigung der Kapazitäten:

$$|Z| = K + 1 = 11$$

Der Zustandsraum sieht folgendermaßen aus:

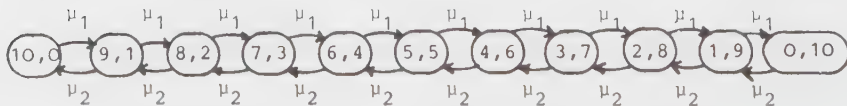


Fig.4.5

Wenn wir die Kapazitäten berücksichtigen, erhalten wir nur den folgenden Teilzustandsraum, der die ausführbaren Zustände und die Blockierzustände (mit * gekennzeichnet) (die unmittelbaren Nachbarn der ausführbaren Zustände) beinhaltet:

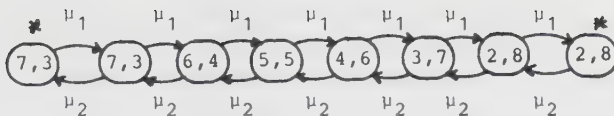


Fig.4.6

Nach dem Theorem können wir ein äquivalentes geschlossenes Netz ohne Blockierung entwickeln, das als die folgende Gesamtanzahl der Aufträge $\hat{K}$ enthält:

$$\hat{K} = |Z| - 1 = 7$$

Damit können wir den Zustandsraum mit $\hat{K}=7$ für das geschlossene Netz ohne Blockierung zeichnen:

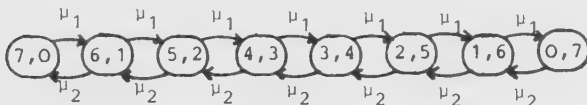


Fig. 4.7

Wie wir sehen, entspricht dieser Zustandsraum Fig.4.7 genau dem in Fig.4.6. Da dieses Netz die Produktformbedingungen erfüllt, können wir den Algorithmus TUPAL (Gl.A1.3) anwenden und die Durchsatzgrößen $\lambda(\hat{K})$ wie folgt berechnen:

$$\begin{aligned} \lambda(1) &= 0.344827 & \lambda(4) &= 0.488511 \\ \lambda(2) &= 0.438728 & \lambda(5) &= 0.494883 \\ \lambda(3) &= 0.473869 & \lambda(6) &= 0.497712 \\ \lambda(7) &= 0.498972 & & \end{aligned}$$

Der Durchsatz $\lambda(7)$ entspricht genau dem Durchsatz $\lambda(10)$ im Blockiernetz. Mit Hilfe dieser Werte und Benutzung der Gl.(4.7) können wir die Zustandswahrscheinlichkeiten, die wiederum genau den Teilzuständen im Blockiernetz entsprechen, wie folgt bestimmen:

$$\begin{aligned}
 p(7,0) &= 0.5509 &= p(7,3)^{\wedge} \\
 p(6,1) &= 0.248 &= p(7,3)^{\wedge\wedge} \\
 p(5,2) &= 0.111 &= p(6,4) \\
 p(4,3) &= 0.05 &= p(5,5) \\
 p(3,4) &= 0.0226 &= p(4,6) \\
 p(2,5) &= 0.01 &= p(3,7) \\
 p(1,6) &= 0.00457 &= p(2,8)^{\wedge\wedge} \\
 p(0,7) &= 0.00205 &= p(2,8)^{\wedge}
 \end{aligned}$$

wobei $p(7,3)^{\wedge}$ die Blockierwahrscheinlichkeit der 1. Station und $p(2,8)^{\wedge}$ die Blockierwahrscheinlichkeit der 2. Station ist.

Aus den ermittelten Zustandswahrscheinlichkeiten können wir dann die mittlere Anzahl von Aufträgen in der i-ten Station im Blockiernetz mit Gl.(4.8) berechnen, wobei wir darauf achten müssen, daß

$$p(7,3) = p(7,3)^{\wedge\wedge} + p(7,3)^{\wedge} = 0.7989$$

bzw.

$$p(2,8) = p(2,8)^{\wedge\wedge} + p(2,8)^{\wedge} = 0.00662$$

ist.

Damit erhalten wir die mittlere Anzahl von Aufträgen

$$\bar{k}_1 = \underline{6.646} \qquad \bar{k}_2 = \underline{3.354}$$

Mit Hilfe des Littleschen Gesetzes berechnen wir die mittleren Verweilzeiten der Aufträge in den Stationen wie folgt:

$$\bar{t}_1 = \frac{\bar{k}_1}{\lambda_1} = \underline{13.29} \qquad \bar{t}_2 = \underline{6.722}$$

Die Auslastung:

$$\rho_1 = \frac{\lambda_1}{\mu_1} = \underline{0.9979} \qquad \rho_2 = \underline{0.449}$$

Die Blockierwahrscheinlichkeiten:

$$P_{B_1} = 1 - \rho_1 = \underline{0.0021} \qquad P_{B_2} = 1 - \rho_2 = \underline{0.5509}$$

Man sieht leicht, daß die Blockierzustände $p(7,3)^{\wedge}$ und $p(2,8)^{\wedge}$ die gleichen Werte liefern.

Die mittleren Blockierzeiten der Prozesse in den einzelnen Stationen sind

$$\bar{t}_{B_1} = 0$$

$$\bar{t}_{B_2} = \underline{5,0887}$$

In der folgenden Tabelle stelle ich diese Werte den Simulationswerten gegenüber. Man kann leicht feststellen, daß die Ergebnisse exakt sind. OKAP bedeutet, daß das Netz ohne Berücksichtigung der Kapazitäten analysiert worden ist.

	Analytisch	Simulation	Abw.	OKAP
$\bar{k}_1$	6.646	6.649	0.045 %	9.184
$\bar{k}_2$	3.354	3.351	0.089 %	0.817
$\bar{t}_1$	13.290	13.340	0.374 %	18.370
$\bar{t}_2$	6.722	6.724	0.029 %	1.633
λ	0.498	0.499	—	0.499

FABLO (Faltungsalgorithmus für Blockiernetze)

Ein effizienter Algorithmus für 2-Stationen-Netze mit Blockierung

Mit den bisherigen Überlegungen können wir ohne Umweg einen effizienten Algorithmus für die Analyse der Blockiernetze angeben, der exakte Ergebnisse liefert.

Der Algorithmus basiert auf der Berechnung der Normalisierungskonstanten. Um diese Größe in Blockiernetzen zu berechnen, modifiziere ich den Faltungsalgorithmus von /CHAN 75a/ wie folgt:

$$G = G_1 \otimes G_2$$

wobei

$$G_i = \begin{bmatrix} x_i(0) \\ x_i(1) \\ . \\ . \\ x_i(K) \end{bmatrix}$$

mit $x_i(0) = 1$

$$x_i(k) = \begin{cases} x_i^k & \text{für } k = 1, \dots, M_i+1 \\ 0 & k = M_i+2, \dots, K \end{cases}$$

Der Durchsatz läßt sich dann aus der Normalisierungskonstanten ermitteln:

$$\lambda(K) = \frac{G(M_1+M_2-K+N-1)}{G(M_1+M_2-K+N)} \quad (4.17)$$

Die mittlere Anzahl von Aufträgen in der i-ten Station ergibt sich aus:

$$\bar{k}_i(K) = \frac{1}{G(K)} \left[\begin{array}{c} M_i \\ \sum_{l=K-M_j}^1 x_i^l \cdot x_j^{K-l} \\ + M_i x_i^{M_i+1} x_j^{K-M_i-1} \\ + (K-M_j) x_i^{K-M_j-1} x_j^{M_j+1} \end{array} \right] \quad (4.18)$$

für $i, j = 1, 2 \quad i \neq j$

Die anderen Leistungsgrößen können dann aus den entsprechenden Formeln (Gln.(4.11-4.15)) bestimmt werden.

BEWERTUNG

Zur Validierung dieser Methode wurden zahlreiche Beispiele durchgeführt und die Ergebnisse mit der Simulation verglichen. Die Methode liefert exakte Ergebnisse. Im folgenden möchte ich Ergebnisse von 5 Beispielen zeigen und im Gesamtkapitel 5 diese Beispiele als Basis für den Vergleich der Methoden annehmen.

Beispiel 1

$K = 20$ Aufträge $M_1 = 18$ $M_2 = 10$ Kapazitäten
 $1/\mu_1 = 2$ $1/\mu_2 = 3.333$ (mittlere Bedienzeiten)

$$\begin{aligned} \hat{K} &= M_1 + M_2 - K + N = 10 & (\text{nach Gl. (4.6)}) \\ \lambda(\hat{K}) &= \lambda(K) = 0.2992 \end{aligned}$$

$\rho_1 = 0.5985$	$\rho_2 = 0.9975$	(nach Gl. (4.12))
$P_{B1} = 0.4014$	$P_{B2} = 0.0024$	(nach Gl. (4.14))
$\bar{k}_1 = 10.8589$	$\bar{k}_2 = 9.1412$	(nach Gl. (4.18))
$\bar{t}_1 = 36.2846$	$\bar{t}_2 = 30.5441$	(nach Gl. (4.11))

Beispiel 2

$$K = 25$$

$$M_1 = 18$$

$$M_2 = 13$$

$$1/\mu_1 = 0.8$$

$$1/\mu_2 = 0.5$$

$$\hat{K} = M_1 + M_2 - K + N = 8$$

$$\lambda(\hat{K}) = \lambda(K) = 1.2389$$

$$\rho_1 = 0.9911$$

$$\rho_2 = 0.6194$$

$$P_{B1} = 0.0088$$

$$P_{B2} = 0.3805$$

$$\bar{k}_1 = 17.0945$$

$$\bar{k}_2 = 7.9054$$

$$\bar{t}_1 = 13.7978$$

$$\bar{t}_2 = 6.3808$$

Beispiel 3

$$K = 15$$

$$M_1 = 10$$

$$M_2 = 10$$

$$1/\mu_1 = 1.1111$$

$$1/\mu_2 = 1.5$$

$$\hat{K} = M_1 + M_2 - K + N = 7$$

$$\lambda(\hat{K}) = \lambda(K) = 0.6434$$

$$\rho_1 = 0.7148$$

$$\rho_2 = 0.965$$

$$P_{B1} = 0.2851$$

$$P_{B2} = 0.034$$

$$\bar{k}_1 = 6.3099$$

$$\bar{k}_2 = 8.691$$

$$\bar{t}_1 = 9.8070$$

$$\bar{t}_2 = 13.506$$

Beispiel 4

$$K = 8$$

$$M_1 = 4$$

$$1/\mu_1 = 10$$

$$M_2 = 7$$

$$1/\mu_2 = 2.5$$

$$\hat{K} = 5$$

$$\lambda(\hat{K}) = \lambda(K) = 0.0999$$

$$\rho_1 = 0.9992$$

$$P_{B1} = 0.0007$$

$$\bar{k}_1 = 3.9186$$

$$\bar{t}_1 = 39.2155$$

$$\rho_2 = 0.2498$$

$$P_{B2} = 0.7501$$

$$\bar{k}_2 = 4.0813$$

$$\bar{t}_2 = 40.8431$$

Beispiel 5

$$K = 50$$

$$M_1 = 31$$

$$1/\mu_1 = 4$$

$$M_2 = 24$$

$$1/\mu_2 = 2$$

$$\hat{K} = 7$$

$$\lambda(\hat{K}) = \lambda(K) = 0.249$$

$$\rho_1 = 0.996$$

$$P_{B1} = 0.004$$

$$\bar{k}_1 = 30.533$$

$$\bar{t}_1 = 122.614$$

$$\rho_2 = 0.498$$

$$P_{B2} = 0.502$$

$$\bar{k}_2 = 19.466$$

$$\bar{t}_2 = 78.173$$

Die Tabelle auf der nächsten Seite zeigt die Simulationsergebnisse.

Simulation der Beispiele

Nr.	K	M_1	M_2	$1/\mu_1$	$1/\mu_2$	τ_1	τ_2	K_1	K_2	λ	ρ_1	ρ_2
1	20	18	10	2.000	3.333	36.58	29.84	11.02	8.987	0.3013	0.6025	0.9038
2	25	18	13	0.800	0.500	13.85	6.383	17.12	7.884	1.235	0.9889	0.6176
3	15	10	10	1.111	1.500	9.803	13.27	6.369	8.631	0.6504	0.7219	0.9745
4	8	4	7	10.00	2.500	39.82	41.34	3.925	4.075	0.0986	0.9857	0.2464
5	50	31	24	4.000	2.000	124.4	79.16	30.57	19.43	0.2457	0.9830	0.4915

Die nachstehende Tabelle zeigt die Abweichungen (in %) der berechneten Lösungen von den simulierten Werten.

$$\Delta = \frac{|\text{Simulationswert} - \text{berechneter Wert}|}{\text{Simulationswert}} \cdot 100$$

	Δ in %				
Nr.	$\bar{k}_1$	$\bar{k}_2$	$\bar{t}_1$	$\bar{t}_2$	λ
1	1.45	1.71	0.82	2.35	0.673
2	0.18	0.27	0.36	0.03	0.315
3	0.93	0.68	0.04	1.81	1.076
4	0.15	0.15	1.51	1.21	1.345
5	0.13	0.21	1.45	1.25	1.343

Die geringen Abweichungen gegenüber der Simulation rühren von den verwendeten Zufallszahlengeneratoren her.

4.2. Zustandsraumtransformationmethode für allgemeine geschlossene Warteschlangennetze mit Blockierung

In diesem Abschnitt werde ich das Konzept der Zustandsraumtransformationmethode auf allgemeine geschlossene Netze mit Blockierung erweitern. Es gelten alle Annahmen, die zu Beginn des Kapitels 4 gemacht wurden.

Um für ein gegebenes geschlossenes Blockiernetz ein äquivalentes Nichtblockiernetz zu entwickeln, müssen wir wie in Abschnitt 4.1 die Gesamtzahl der Aufträge $\hat{K}$ im Nichtblockiernetz bestimmen. Diese können wir aus Gl.(4.1) ermitteln.

Da aber die Anzahl der Zustände für große Netze enorm groß ist, können wir nicht immer den Zustandsraum zeichnen, daraus die nicht ausführbaren Zustände eliminieren und die Anzahl der ausführbaren Zustände und deren unmittelbare Nachbarn zählen, um die Anzahl der Zustände $|Z|$ zu bestimmen und daraus dann mit Gl.(4.1) $\hat{K}$ zu ermitteln.

Dafür habe ich folgende Vorgehensweise entwickelt, die die Anzahl der möglich ausführbaren Zustände und deren Nachbarn liefert. Als Grundlage nehme ich wieder den Faltungsalgorithmus von /CHAN 75a/.

LEMMA

Die Anzahl der ausführbaren Zustände und deren Nachbarn ergibt sich im geschlossenen Warteschlangennetz mit Blockierung aus der folgenden Formel:

$$Z = Z_1 \otimes Z_2 \otimes \dots \otimes Z_N \quad (4.19)$$

wobei Z_i ($i=1, \dots, N$) ein $(K+1)$ -dimensionaler Vektor ist.

$$Z_i = \begin{bmatrix} a_i(0) \\ a_i(1) \\ \vdots \\ a_i(K) \end{bmatrix} \quad (4.20)$$

mit den Elementen

$$a_i(k) = \begin{cases} 1 & k = 0, 1, \dots, M_i+1 \\ \text{für} & \\ 0 & k = M_i+2, \dots, K \end{cases}$$

Damit ergibt sich die Anzahl $|Z|$ der ausführbaren Zustände aus dem letzten Element $a(K)$ des Ergebnisvektors Z . Aus Gl.(4.1) können wir dann die Gesamtanzahl $\hat{K}$ der Aufträge im äquivalenten Netzwerk ohne Blockierung bestimmen.

Hier stellt man fest, daß die Abbildung des Zustandsraums des Blockiernetzes auf den Zustandsraum des äquivalenten Netzes ohne Blockierung nicht bijektiv ist, d.h. die Anzahl der Zustände in beiden Räumen stimmt nicht überein, d.h. die Zustände können nicht wie bei Netzen mit 2-Stationen eindeutig 1:1 abgebildet werden. Damit können aus Gl.(4.1) reelle Werte für $\hat{K}$ ermittelt werden. Um dies zu vermeiden, wird $\hat{K}$ als derjenige Integerwert angenommen, der am nächsten bei der ermittelten Zustandszahl liegt.

Zur Verdeutlichung dieser Überlegungen möchte ich an dieser Stelle ein numerisches Beispiel geben:

Beispiel:

Gegeben ist ein Netz mit $K=5$ Aufträgen und $N=3$ Stationen. Die Stationen haben endliche Kapazitäten $M_i=2$ für $i=1,2,3$. Die mittleren Bedienzeiten sind $1/\mu_1=1$, $1/\mu_2=0.9$, $1/\mu_3=2$. Die Stationen sind seriell geschaltet.

Zunächst ermitteln wir die Anzahl der ausführbaren Zustände mit Nachbarn aus Gl.(4.19):

$$Z = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 6 \\ 10 \\ 12 \\ 12 \end{bmatrix}$$

Damit erhalten wir $|Z| = 12$.

Wenn wir den Zustandsraum für dieses gegebene Netz zeichnen, stellen wir fest, daß die Anzahl der ausführbaren Zuständen mit den unmittelbaren Nachbarn tatsächlich das gleiche Ergebnis liefert wie oben:

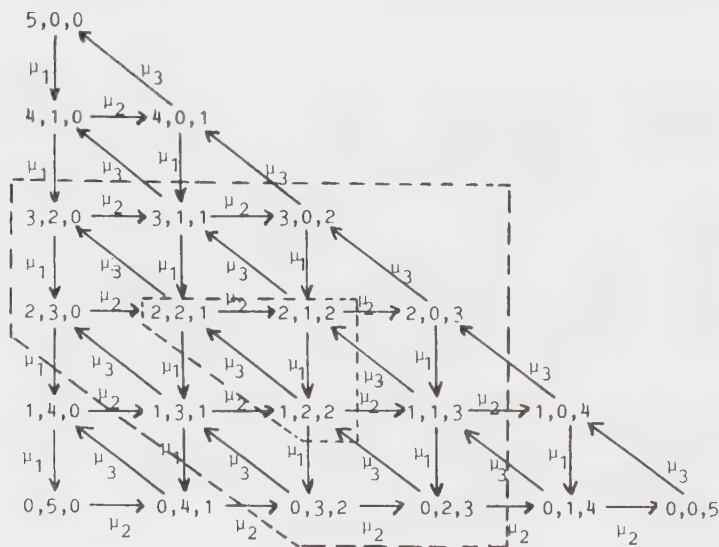


Fig. 4.8

Die Gesamtzahl $\hat{K}$ der Aufträge im äquivalenten Nichtblockier-netz ermitteln wir aus Gl.(4.1) wie folgt:

$$|Z| = 12 = \binom{\hat{K} + 2}{2}$$

Für $\hat{K}=3$ existieren 10 Zustände und für $\hat{K}=4$ 15 Zustände. Da 10 Zustände näher zu 12 liegen, wird die Gesamtanzahl $\hat{K}$ der Aufträge im äquivalenten Netz in diesem Fall als $\hat{K}=3$ angenommen.

Wenn wir den Zustandsraum für $\hat{K}=3$ im äquivalenten Netz zeichnen, sehen wir, daß weniger Zustände existieren als im Blockiernetz und dadurch keine eindeutige Abbildung der Zustände möglich ist.

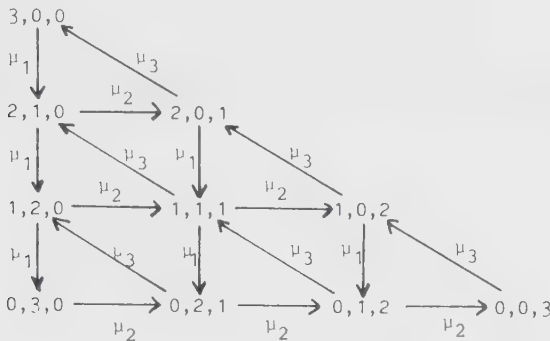


Fig. 4.9

Daraus folgt, daß die durch diese Vorgehensweise ermittelten Leistungsgrößen im Falle allgemeiner geschlossener Netze nicht exakt sein können.

Wie in Abschnitt 4.1 wird zunächst das äquivalente Netzwerk ohne Blockierung mit der ermittelten Gesamtzahl $\hat{K}$ der Aufträge mit dem TUPAL-Algorithmus analysiert und der Durchsatz $\lambda(\hat{K})$ berechnet.

Der Durchsatz $\lambda(\hat{K})$ des äquivalenten Netzwerks entspricht nahezu exakt dem Durchsatz $\lambda(K)$ des Blockiernetzes mit der Gesamtzahl $\hat{K}$ der Aufträge.

Aus dem Durchsatz $\lambda(K)$ können dann die Durchsätze der einzelnen Stationen aus

$$\lambda_i(K) = e_i \lambda(K) \quad \text{für } i=1, \dots, N \quad (4.21)$$

Die Auslastung der Stationen und ihre Blockierwahrscheinlichkeiten können mit den Gln.(4.12) bzw. (4.14) ermittelt werden.

Man kann auch die Gesamtverweilzeit der Aufträge im System wie folgt ermitteln:

$$t_G = \frac{K}{\lambda(K)} \quad (4.22)$$

Damit kann auch die Gesamtblockierzeit der Aufträge im System leicht berechnet werden:

$$t_{BG} = t_G^* - t_G \quad (4.23)$$

wobei t_G^* die Gesamtverweilzeit des Systems ohne Berücksichtigung der Kapazitäten ist.

Um die mittlere Anzahl von Aufträgen $\bar{k}_i$ in den einzelnen Stationen zu berechnen, hatten wir in Abschnitt 4.1 zunächst die Wahrscheinlichkeiten der Zustände im äquivalenten Netz berechnet und die Resultate auf die Zustände des Blockiernetzes abgebildet. Durch die so ermittelten Zustandswahrscheinlichkeiten konnten wir die mittlere Anzahl von Aufträgen mit Gl.(4.8) im Tandemnetz exakt berechnen. Da wir hier keine bijektive Abbildung haben, können wir die Vorgehensweise der Analyse eines Tandem-Netzes nicht fortsetzen, um die mittlere Anzahl von Aufträgen $\bar{k}_i$ in der i -ten Station zu berechnen.

Bekanntlich gilt Gl.(4.8) für jede Art von Warteschlangennetzen, um die Größe $\bar{k}_i$ zu berechnen. Das Problem liegt nun in der Bestimmung der Zustandswahrscheinlichkeiten, von denen dann die Randwahrscheinlichkeiten abgeleitet werden können. Wenn wir die Zustandswahrscheinlichkeiten über Produktformlösungen mit (wie im Abschnitt 4.4) oder ohne Berücksichtigung der Stationskapazitäten bestimmen (Gl.(A1.4)), erhalten wir Abweichungen von den tatsächlichen Werten.

Für die Bestimmung der Randwahrscheinlichkeiten und daraus dann mit Gl.4.8 der mittleren Anzahl von Aufträgen $\bar{k}_i$ in der i -ten Station habe ich die folgende Überlegung realisiert und sie Projektionsmethode genannt.

PROJEKTIONSMETHODE

Ausgangspunkt dieses Algorithmus ist das Produktformnetz, das wir erhalten, wenn wir die Kapazitätsbeschränkungen der Stationen zunächst aufgeben. Für dieses Produktformnetz werden alle Zustandswahrscheinlichkeiten aus Gl.(A1.1). berechnet. Dann müssen die aufgrund der Blockierungen nicht ausführbaren Zustände normalisiert werden, und zwar in dem Sinne, daß diese Zustände den Kapazitäten der Stationen angepaßt werden.

Die Normalisierung der Zustände wird durch die folgende Formel durchgeführt:

Wenn die Anzahl der Aufträge in der i-ten Station die Kapazität überschreitet, dann wird die Auftragsverteilung auf der folgenden Formel solange normalisiert, bis keine Verletzung der Kapazitäten mehr vorliegt:

$$v_j \quad k_j := \begin{cases} M_j & j=i \\ \text{für} & \\ k_j + (k_i - M_i) p_{ji} \frac{e_i}{e_i (1 - p_{ii})} & j \neq i \end{cases} \quad (4.24)$$

wobei p_{ji} die Übergangswahrscheinlichkeit der Aufträge von der j-ten zur i-ten Station ist.

e_j ist die mittlere Besuchshäufigkeit eines Auftrags in der j-ten Station

Gl.(4.24) besagt, daß die über der Kapazität liegenden Aufträge in einer Station auf die ihr vorgeschalteten Stationen verteilt werden. Dies erfolgt gemäß dem Anteil, den die vorgeschalteten Stationen am Zufluß zur übergelaufenen Station haben (vgl. Gl.(A1.2)).

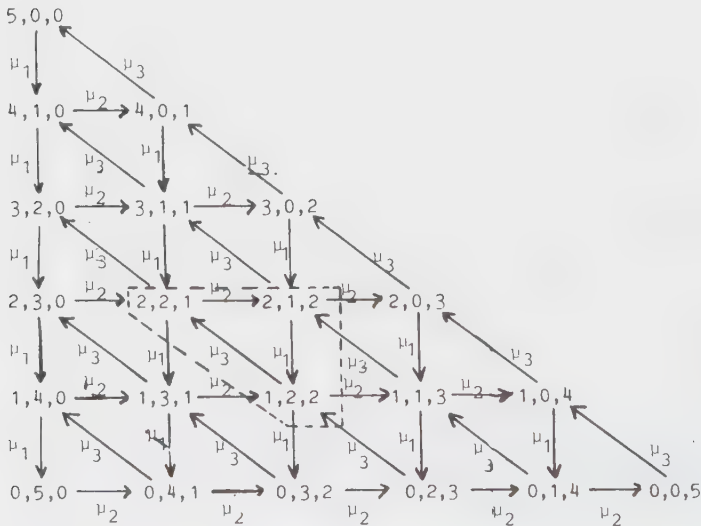
Für jeden nicht-ausführbaren Zustand des Netzes ohne Berücksichtigung der Kapazitäten wird die Normalisierung so lange durchgeführt, bis keine Kapazitätsüberschreitungen mehr vorkommen.

Zur Verdeutlichung des Konzeptes möchte ich an dieser Stelle

das gleiche Beispiel fortsetzen, das wir in diesem Abschnitt eingeführt haben.

Wir hatten den Zustandsraum in Figur 4.8 für das gegebene Netz ohne Blockierung erhalten. Zunächst werden aus Gl.(A1.1) die Werte aller Zustandswahrscheinlichkeiten berechnet. Wenn wir die Kapazitäten der Stationen betrachten, können wir nicht ohne weiteres aus Gl.(4.8) die mittlere Anzahl von Aufträgen $\bar{k}_i$ in den jeweiligen Stationen ermitteln, sondern müssen die Tatsache berücksichtigen, daß im Blockiernetz die Stationen nur jeweils max. 2 Aufträge aufnehmen können, d.h. es sind die Zustände (2,2,1), (2,1,2) und (1,2,2) ausführbar. Alle anderen nicht-ausführbaren Zustände müssen normalisiert werden.

Die folgende Figur 4.10. zeigt den Zustandsraum für das Netz ohne Berücksichtigung der Kapazitäten.



Figur 4.10.

Die ausführbaren Zustände sind in Fig.4.10 umrandet. Die nicht-ausführbaren Zustände müssen wie folgt normalisiert werden.

Betrachten wir zunächst den nicht-ausführbaren Zustand

(4,0,1). Da sich in diesem Beispiel nur max. 2 Aufträge in jeder Station befinden dürfen, müssen wir die Anzahl der Aufträge an die Kapazitäten der Stationen anpassen, d.h.:

Da die Anzahl 4 der Aufträge in der 1. Station über der Kapazität liegt, setzen wir mit Gl.(4.24):

$$k_1 = \underline{2}$$

Die restlichen 2 Aufträge müssen auf die anderen Stationen verteilt werden. Mit Gl.(4.24) erhalten wir:

$$k_2 := k_2 + (k_1 - M_1) \cdot p_{21} \cdot \frac{e_2}{e_1}$$

Da $p_{21}=0$ ist, folgt dann

$$k_2 = \underline{0}$$

Für die 3. Station

$$k_3 := k_3 + (k_1 - M_1) \cdot p_{31} \cdot \frac{e_3}{e_1} = 3$$

Durch die Normalisierung des Zustands (4,0,1) wurde der Zustand (2,0,3) erhalten, wobei leicht ersichtlich ist, daß in diesem Fall die Kapazität der 3. Station überschritten ist und der Zustand (2,0,3) weiter normalisiert werden muß.

Wir setzen diesmal

$$k_3 = \underline{2}$$

Für die 2. Station

$$k_2 := k_2 + (k_3 - M_3) \cdot p_{23} \cdot \frac{e_2}{e_3} = 1$$

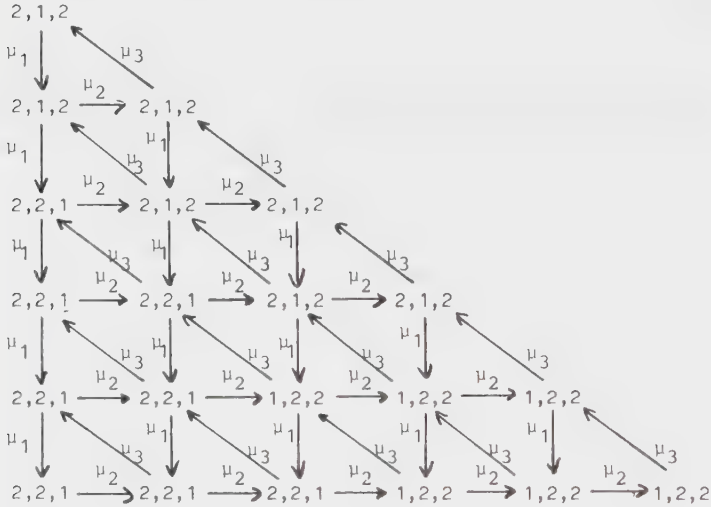
Für die 1. Station

$$k_1 := k_1 + (k_3 - M_3) \cdot p_{13} \cdot \frac{e_1}{e_3} = \underline{2}$$

Nach der zweiten Normalisierung erhalten wir den Zustand

(2,1,2) und stellen fest, daß die Anzahl der Aufträge in jeder Station den Kapazitäten angepaßt sind:

Durch diese Vorgehensweise erhalten wir dann den folgenden Zustandsraum für das Blockiernetz.



Figur 4.11

m.a.W.: Wir haben den Zustandsraum vom Netz ohne Kapazitätsbeschränkung auf den Zustandsraum des äquivalenten Netzes mit Blockierung projiziert. Die aus dem Netz ohne Berücksichtigung der Kapazitäten ermittelten Werte für die Zustandswahrscheinlichkeiten werden den entsprechenden normalisierten Zuständen zugewiesen.

Aus den so ermittelten Zustandswahrscheinlichkeiten $p(k)$ können leicht die Randwahrscheinlichkeiten und daraus dann mit Gl.(4.8) die mittlere Anzahl von Aufträgen $\bar{k}_i$ in den Stationen bestimmt werden. Die anderen Leistungsgrößen wie die mittlere Verweilzeit $\bar{t}_i$ aus Gl.(4.11) und die mittlere Blockierzeit $\bar{t}_B$ aus Gl.(4.15) können auch leicht berechnet werden.

Beispiel

Wir erhalten hier die Leistungsgrößen für das Beispiel, das wir zu Beginn des Abschnitts 4.2. beschrieben haben. Wir hatten im Laufe des Abschnitts festgestellt, daß die Anzahl der ausführbaren Zustände mit den Nachbarn 12 beträgt, woraus dann die Gesamtanzahl $\hat{K}$ der Aufträge im Netz ohne Blockierung mit $\hat{K}=3$ ermittelt wurde.

Mit Hilfe des TUPAL können wir den Durchsatz $\lambda(\hat{K})$ im Nicht-blockiernetz wie folgt berechnen:

$$\lambda(K) \approx \lambda(\hat{K}) = 0.43$$

der dem Durchsatz des Netzes mit Blockierung entspricht. Die anderen interessanten Größen können aus den entsprechenden Gleichungen (Gl.(4.11) - (4.15)) berechnet werden.

Um aber die mittlere Anzahl von Aufträgen $\bar{k}_i$ in der i-ten Station zu berechnen, müssen wir die Projektionsmethode anwenden.

Zunächst berechnen wir die Zustandswahrscheinlichkeiten im Netz ohne Berücksichtigung der Kapazitäten:

$p(5,0,0) = 0.009032$	$p(3,0,2) = 0.0361$
$p(4,1,0) = 0.008129$	$p(2,1,2) = 0.0325$
$p(3,2,0) = 0.007316$	$p(1,2,2) = 0.0292$
$p(2,3,0) = 0.006584$	$p(0,3,2) = 0.0263$
$p(1,4,0) = 0.00592$	$p(2,0,3) = 0.0722$
$p(0,5,0) = 0.00533$	$p(1,1,3) = 0.065$
$p(4,0,1) = 0.018$	$p(0,2,3) = 0.0585$
$p(3,1,1) = 0.01625$	$p(1,0,4) = 0.1445$
$p(2,2,1) = 0.01463$	$p(0,1,4) = 0.13$
$p(1,3,1) = 0.01317$	$p(0,0,5) = 0.289$
$p(0,4,1) = 0.0118$	

Wie aus der Figur 4.11 ersichtlich, normalisieren wir die nicht-ausführbaren Zustände und erhalten einen Zustandsraum für

Blockiernetze mit den Kapazitäten angepaßten Zuständen:

$$p(2,2,1)^{\wedge} = p(3,2,0) + p(2,3,0) + p(1,4,0) + p(1,3,1) + p(0,5,0) + p(0,4,1) + p(2,2,1) = \underline{0.06463}$$

$$p(2,1,2)^{\wedge} = p(5,0,0) + p(4,1,0) + p(4,0,1) + p(3,1,1) + p(3,0,2) + p(2,0,3) + p(2,1,2) = \underline{0.192261}$$

$$p(1,2,2)^{\wedge} = p(1,2,2) + p(1,1,3) + p(1,0,4) + p(0,3,2) + p(0,2,3) + p(0,1,4) + p(0,0,5) = \underline{0.7425}$$

Die mittlere Anzahl von Aufträgen kann daraus dann wie folgt berechnet werden:

$$\bar{k}_1 = \sum_{l=1}^{M_1} l * p_i(l) = \underline{1.256} \quad \bar{k}_2 = \underline{1.807} \quad \bar{k}_3 = \underline{1.934}$$

Die mittlere Verweilzeit ergibt sich dann aus Gl.(4.11):

$$\bar{t}_1 = \underline{2.92} \quad \bar{t}_2 = \underline{4.20} \quad \bar{t}_3 = \underline{4.49}$$

In der folgenden Tabelle vergleiche ich diese Werte mit den Ergebnissen denen der Simulation:

	Analytisch	Simulation	Abw.
$\bar{k}_1$	1.256	1.297	3.26
$\bar{k}_2$	1.807	1.832	1.4
$\bar{k}_3$	1.934	1.871	3.25
$\bar{t}_1$	2.92	2.921	-
$\bar{t}_2$	4.20	4.129	1.70
$\bar{t}_3$	4.49	4.217	6.08
λ	0.43	0.44	2.27

Bemerkung

Bei Netzen mit nicht-seriell geschalteten Stationen kann die Normalisierung der Zustände mit Gl.(4.24) wegen der Übergangswahrscheinlichkeiten nicht-natürliche Werte für die Anzahl k_i der

Aufträge in den jeweiligen Stationen liefern. Diese Auftragszahlen werden gleichfalls bei der Ermittlung der mittleren Auftragszahlen zur Gewichtung der Zustandswahrscheinlichkeiten für das Blockiernetz herangezogen.

Zur Verdeutlichung dieses Konzeptes möchte ich im folgenden Beispiel ein Central-Server-Modell mit beschränkten Kapazitäten analysieren.

Beispiel

Gegeben ist ein Central-Server-Modell mit $N=3$ Stationen und $K=3$ Aufträgen. Die Kapazitäten der Stationen sind wie folgt gegeben:

$$M_1 = 2 \quad M_2 = 1 \quad M_3 = 1$$

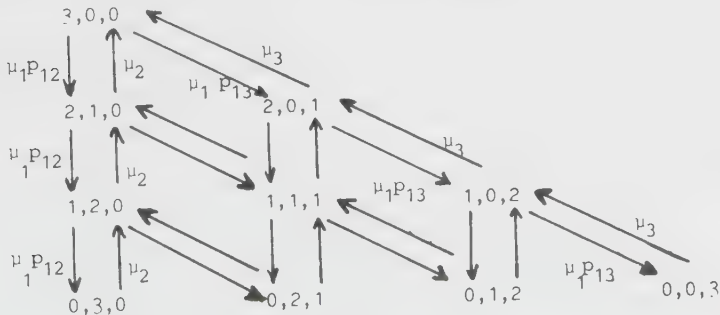
Die Bedienzeiten der Stationen sind exponentiell verteilt mit den folgenden Mittelwerten:

$$1/\mu_1 = 1 \quad 1/\mu_2 = 1.25 \quad 1/\mu_3 = 2$$

Die Übergangswahrscheinlichkeiten betragen

$$p_{12} = 0.6 \quad p_{13} = 0.4 \quad p_{21} = p_{31} = 1$$

Das Netzwerk ohne Berücksichtigung der Kapazitäten hat den folgenden Zustandsraum:



Figur 4.12.

Dieses Netz hat Produktformlösung (Gl.(A1.1)), wobei wir für die Zustandswahrscheinlichkeiten die folgenden Werte erhalten:

$$p(3,0,0) = 0.1609$$

$$p(2,1,0) = 0.1206$$

$$p(1,2,0) = 0.0905$$

$$p(0,3,0) = 0.0679$$

$$p(2,0,1) = 0.1287$$

$$p(1,1,1) = 0.0965$$

$$p(0,2,1) = 0.0734$$

$$p(1,0,2) = 0.1030$$

$$p(0,1,2) = 0.0772$$

$$p(0,0,3) = 0.0824$$

Um die nicht-ausführbaren Zustände zu normalisieren, müssen wir die Anzahl der Aufträge an die Kapazitäten anpassen.

Der Zustand (3,0,0) wird mit Gl.(4.24) wie folgt normalisiert:

$$k_1 := M_1 + (k_1 - M_1) \cdot p_{11} \cdot \frac{e_1}{e_1} = 2$$

$$k_2 := k_2 + (k_1 - M_1) \cdot p_{21} \cdot \frac{e_2}{e_1} = 0.6$$

$$k_3 := k_3 + (k_1 - M_1) \cdot p_{31} \cdot \frac{e_3}{e_1} = 0.4$$

Da nun keine Auftragszahl mehr die Kapazität überschreitet, ist die Normalisierung dieses Zustands beendet. Der Zustand (3,0,0) wird im äquivalenten Netz durch den Zustand (2,0.6,0.4) ersetzt.

Die Normalisierung z.B. des Zustands (1,2,0) wird wie folgt durchgeführt:

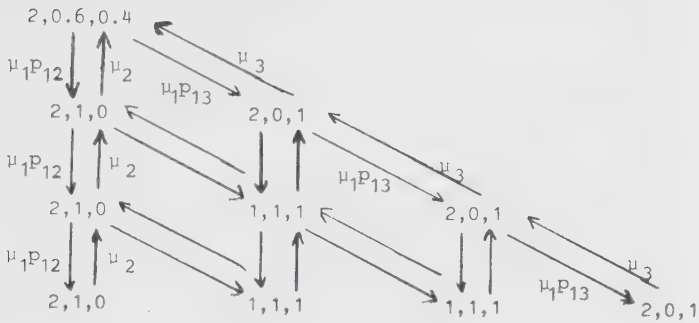
$$k_1 := k_1 + (k_2 - M_2) \cdot p_{12} \cdot \frac{e_1}{e_2} = 2$$

$$k_2 := k_2 + (k_2 - M_2) \cdot p_{22} \cdot \frac{e_2}{e_1} = 1$$

$$k_3 := k_3 + (k_2 - M_2) \cdot p_{32} \cdot \frac{e_3}{e_1} = 0$$

Der normalisierte Zustand lautet demnach (2,1,0). Die weiteren Normalisierungen erfolgen analog.

Wir erhalten schließlich das folgende äquivalente Netz:



Figur 4.13.

Zur Ermittlung der mittleren Anzahl von Aufträgen $\bar{k}_i$ in den Stationen werden die Wahrscheinlichkeiten nun mit den Auftragsverteilungen aus dem äquivalenten Zustandsraum gewichtet:

$$\begin{aligned}\bar{k}_1 &= 2 p(3,0,0) + 2 p(2,1,0) + 2 p(1,2,0) + 2 p(0,3,0) + \\ &\quad 2 p(2,0,1) + 2 p(1,0,2) + 2 p(0,0,3) + 1 p(1,1,1) + \\ &\quad 1 p(0,2,1) + 1 p(0,1,2) \\ &= \underline{1.7539}\end{aligned}$$

$$\begin{aligned}\bar{k}_2 &= 0.6 p(3,0,0) + 1 p(2,1,0) + 1 p(1,2,0) + 1 p(0,3,0) + \\ &\quad 1 p(1,1,1) + 1 p(0,2,1) + 1 p(0,1,2) \\ &= \underline{0.62264}\end{aligned}$$

$$\begin{aligned}\bar{k}_3 &= 0.4 p(3,0,0) + 1 p(2,0,1) + 1 p(1,1,1) + 1 p(0,2,1) + \\ &\quad 1 p(0,0,3) \\ &= \underline{0.62556}\end{aligned}$$

Die Gesamtvorgehensweise für die Analyse der geschlossenen Warteschlangennetze mit Blockierung können wir wie folgt zusammenfassen:

1. Bestimme die Anzahl der ausführbaren Zustände mit den Nachbarn aus Gl.(4.19).
2. Bestimme die Anzahl der Aufträge $\hat{K}$ für das Netz ohne Blockierung aus Gl.(4.1).
3. Analysiere das Netz mit $\hat{K}$ Aufträgen mit TUPAL und bestimme den Durchsatz $\lambda(\hat{K})$. Der Durchsatz $\lambda(\hat{K})$ aus dem Netz ohne Blockierung entspricht genau dem Durchsatz $\lambda(K)$ aus dem Netz mit Blockierung.
Berechne die Durchsätze einzelner Stationen aus Gl.(4.21), die Auslastungen aus Gl.(4.12) und die Blockierwahrscheinlichkeiten aus Gl.(4.14).
4. Ermittle alle Zustandswahrscheinlichkeiten aus Gl.(A1.1) ohne Berücksichtigung der Kapazitäten.
5. Normalisiere die nicht-ausführbaren Zustände und ermittle die Zustandswahrscheinlichkeiten für das Blockiernetz und daraus dann die Randwahrscheinlichkeiten.
6. Berechne die mittlere Anzahl von Aufträgen aus Gl.(4.8), die mittlere Verweilzeit aus Gl.(4.11) und die mittlere Blockierzeit aus Gl.(4.15).

Bewertung

Zur Validierung der Methode wurden ca. 75 verschiedene Beispiele auf der Rechenanlage CDC-Cyber 173 am RRZE in Erlangen durchgeführt. Als Vergleich wurden die Simulationsergebnisse, die mit dem Software-Paket GPSS-Fortran /SCHM 84/ auf dem Rechner Telefunken TR 440 ermittelt wurden, herangezogen. Dabei wurde mit Gl.(4.19) die Zustandszahl im Blockiernetz und mit der Binomialformel Gl.(4.1) $\hat{K}$ ermittelt. Der Durchsatz im äquivalenten

Nichtblockiernetz wurde mit dem TUPAL, A1.1.1, berechnet. Die größten festgestellten Abweichungen lagen bei 1-3%. Die Methode darf also als nahezu exakt bezeichnet werden.

Die Projektionsmethode zeigte bei den Tests ein sehr stabiles Verhalten. Eine Abweichung von max. 12% wurde nur bei numerisch kleinen Werten überschritten. Die meisten gemessenen Abweichungen lagen wie im durchgeführten Beispiel bei 1-5%.

Die Projektion der Zustände wird automatisch durchgeführt, was natürlich eine lange Rechenzeit mit sich bringt. Hierbei muß jedoch vermerkt werden, daß der Algorithmus die vergleichsweise riesigen Rechenzeiten der numerischen Methoden und der Simulation bei weitem nicht erreicht.

In Anhang III werde ich Ergebnisse von verschiedenen beliebig gewählten numerischen Beispielen auflisten, die mit der Zustandsraumtransformationsmethode sowie mit MVABIO, die in Abschnitt 4.3 bzw. mit den Produktformmethoden für Blockiernetze, die in Abschnitt 4.4 beschrieben werden, erzielt worden sind. Alle analytischen Ergebnisse werden mit denen der Simulation verglichen.

4.3. MITTELWERTANALYSE FÜR BLOCKIERNETZE

Die bisherigen Lösungsansätze betrachteten alle ausführbaren Zustände und entwickelten daraus ihre Algorithmen. In diesem Abschnitt ändert sich unsere Betrachtungsweise auf das System. Ein Netz mit Blockierung wird als Warteschlangennetz gemäß Fig. 4.14 aufgefaßt, wobei wir in der Fig. 4.14 ein 2-Stationen-Netz betrachten.

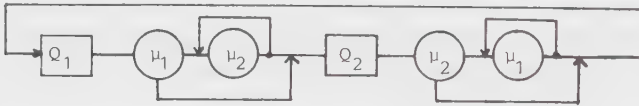


Fig.4.14.

Hinter jeder Bedienstation wird im Bedarfsfalle ein weiterer Abarbeitungszyklus geschaltet. Das Zuschalten der Zusatzbedienphase erfolgt hierbei in Abhängigkeit von der Anzahl der Aufträge in der nachfolgenden Station. Dadurch wird die Bedienzeit des blockierten Auftrags erhöht, während sich die mittlere Verweilzeit in der gefüllten Bedienstation nicht ändert, da ja kein neuer Auftrag in die Warteschlange eingereicht werden kann. Dieser Vorgang muß so lange wiederholt werden, bis in der überfüllten Warteschlange wieder Platz vorhanden ist. Als wichtiges Kriterium tritt hier die Frage nach der Ausschöpfung der Kapazität M_i in den Vordergrund. Nach jeder Erhöhung der Auftragszahl muß überprüft werden, ob die mittlere Anzahl der Aufträge $\bar{k}_i$ diese Grenze überschreitet.

Die Mittelwertanalyse bietet ein gutes Instrument, diesen Vorgang nachzuvollziehen, da die Systemparameter schrittweise mit der Auftragsmenge neu ermittelt werden.

Beginnend mit den üblichen Vorbesetzungen in der MVA, berechnet man Größen wie die mittlere Verweilzeit, Durchsatz und die mittlere Anzahl von Aufträgen. Nach jeder Iteration wird abgefragt, ob die mittleren Auftragszahlen in den Bedienstationen größer sind als die jeweiligen Kapazitäten sind. Sollte das der Fall sein, muß die Iteration in modifizierter Form wiederholt

werden:

Dabei berücksichtigt man, daß in die blockierte Station kein Auftrag mehr hineinkommen kann. Dementsprechend wird die mittlere Verweilzeit dieser Station i verkürzt.

Da die Aufnahmekapazität der "vollen" Bedienstation bei (K-1) Aufträgen im System erschöpft ist, muß der überzählige Auftrag demnach aus der überfüllten Station entfernt werden, d.h. wenn i die volle Bedienstation ist, dann ergibt sich die mittlere Verweilzeit der Aufträge in der Station aus

$$\bar{t}_i(k) = \frac{1}{\mu_i} \left[\bar{k}_i(k-1) + 1 - 1 \right] = \frac{1}{\mu_i} \bar{k}_i(k-1) \quad (4.25)$$

Da dieser Auftrag nicht einfach "verschwinden" kann, muß er sich zwangsläufig in der blockierten Station einfinden. Dort wird er um die mittlere Restbedienzeit $1/\mu_i$ verzögert werden, die Zeit, nach der im Mittel in der ausgelasteten Bedienstation ein Platz frei wird, der Auftrag also weitergegeben werden kann.

Für geschlossene Warteschlangennetze, in denen die Stationen seriell geschaltet sind, bekommt man dann für den obigen Fall:

$$\bar{t}_j(k) = \frac{1}{\mu_j} \left[1 + \bar{k}_j(k-1) \right] + BZ_i = \bar{t}_j(k) + BZ_i \quad (4.26)$$

wobei BZ_i die Bedienzeit der nachfolgenden Station ist.

Im folgenden möchte ich zeigen, warum ein Auftrag sich um die Bedienzeit der nachfolgenden Station länger verweilt.

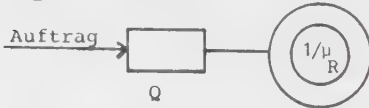
Die blockierende Bedienstation wird an ihrer Weiterarbeit so lange gehindert, bis die folgende Bedienstation einen Auftrag fertig bearbeitet hat. Die Verzögerung der Bearbeitung in der wartenden Bedienstation hängt im Augenblick der Blockierung von der Restbedienzeit des Auftrags in der anschließenden Bedienstation ab.

Wir betrachten ein M/G/1-System mit

$$F_A(t) = 1 - e^{-\lambda t}$$

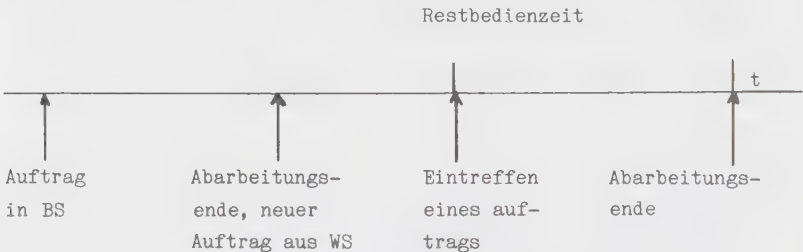
als Ankunftsverteilung und der Bedienzeitverteilung

$$F_B(t) = \text{beliebig}$$



Die Wartezeit w eines gerade eintreffenden Auftrags setzt sich zusammen aus dem Produkt der Warteschlangenlänge Q mal mittlerer Bedienzeit $1/\mu$ plus Restbedienzeit $1/\mu_R$.

Die Restbedienzeit $1/\mu_R$ ist die Zeit, die ein Auftrag in der Bedienstation noch benötigt, um abgearbeitet zu werden, während ein neuer Auftrag ins Wartesystem gelangt.



Die Restbedienzeit $1/\mu_R$ wird für den ankommenden Auftrag zu Null, wenn die Bedienstation inaktiv ist, also leer ist. Dieser Auftrag wird dann sofort in Bearbeitung genommen, ohne zu warten. Für unsere Betrachtungen ist dieser Fall irrelevant, da er in Widerspruch zu der Blockierbedingung steht, daß die Warteschlange voll und ein Auftrag in der Bedienstation sein soll. Es ist daher sinnvoll, hier nur die mittlere Restbedienzeit zu betrachten, wenn das System aktiv ist.

$b(x)$ sei die Dichte der Bedienzeit $x = 1/\mu$.

$b'(x)$ sei die Dichte des Bedienintervalls, in dem ein Auftrag

ankommt.

$$b'(x) \sim x$$

Je länger das Zeitintervall, desto größer ist die Wahrscheinlichkeit, daß ein Auftrag ankommt.

$$b'(x) \sim b(x)$$

Je größer die Wahrscheinlichkeit eines Zeitintervalls x , desto größer ist auch die Wahrscheinlichkeit, daß in diesem Zeitintervall ein Auftrag ankommt.

Da $b'(x)$ Dichtefunktion ist, gilt mit den obigen Definitionen:

$$\int_0^{\infty} b'(x) dx = \Pi \int_0^{\infty} b(x) dx \quad (\Pi \text{ Proportionalitätsfaktor})$$

$$1 = \Pi \cdot \bar{x}$$

$$\Pi = 1/\bar{x}$$

Damit folgt: $b'(x) = x/\bar{x} b(x)$

Die mittlere Länge der Bedienzeit, in der ein Auftrag ankommt, läßt sich somit einfach berechnen.

$$\bar{y} = \int_0^{\infty} x b'(x) dx = 1/\bar{x} \int_0^{\infty} x^2 b(x) dx = \bar{x}^2/\bar{x}$$

Die mittlere Restbedienzeit $1/\mu_R$, die ein Auftrag noch warten muß, ergibt sich zu:

$$1/\mu_R = \bar{x}^2 / 2\bar{x}$$

Setzt man die Momente $\bar{x} = 1/\mu_R$ und $\bar{x}^2 = 2/\mu^2$ für die Exponentialverteilung ein, erhält man für die mittlere Restbedienzeit folgendes Resultat:

$$1/\mu_R = 1/\mu$$

Die Zeit, die ein blockierter Auftrag in der Bedienstation BS_i im Mittel warten muß, beträgt demnach $1/\mu_{(i+1) \bmod 2} + 1$.

Handelt es sich beim untersuchten Netzwerk um ein allgemeines geschlossenes Netz, so teilt sich die Erhöhung der Verweilzeiten um eine Bedienzeit der blockierenden Station auf alle Stationen auf, von denen ein Übergang zur blockierenden Station möglich ist, und zwar gemäß der Übergangswahrscheinlichkeiten. Damit muß also ein Auftrag in der j-ten Station genau die Bedienzeit der vollen i-ten Station gewichtet mit den Übergangswahrscheinlichkeiten p_{ji} länger verweilen. Damit erhalten wir für allgemeine Netze:

$$\begin{aligned}\bar{t}_j(k) &= \frac{1}{\mu_j} \left[1 + \bar{k}_i(k-1) \right] + BZ_i * p_{ji} \\ &= \bar{t}_j(k) + BZ_i * p_{ji}\end{aligned}\tag{4.27}$$

Mit diesen neuen Werten, Gl.n.(4.25,4.26 bzw. 4.27) können $\lambda(k)$, $\bar{k}_i(k)$ neu berechnet werden. Diese Modifikation der Verweilzeiten wird auch bei allen folgenden Iterationen beibehalten. Tritt nun eine zweite Blockierung dieser Station i auf, läßt man bei der Wiederholung der Iteration die Verweilzeit der blockierenden Station konstant, während die Verweilzeit der jeweiligen betroffenen anderen Bedienstationen natürlich ein weiteres mal um eine Bedienzeit der blockierenden Bedienstation erhöht wird.

Den Algorithmus können wir wie folgt zusammenfassen:

1. Schritt Initialwerte

$$\bar{k}_i(0) = 0 \wedge z_i = 1 \wedge BZ_i = 0 \quad \text{für } i=1, \dots, N$$

2. Schritt

- a) Berechne die mittlere Verweilzeit der Aufträge in der i-ten Station

$$\bar{t}_i(k) = \frac{1}{\mu_i} \left[z_i + \bar{k}_i(k-1) \right] + BZ_i \tag{4.28}$$

b) Berechne den Durchsatz und die mittlere Anzahl von Aufträgen

$$\lambda(k) = \frac{k}{\sum_{i=1}^N e_i \cdot \bar{t}_i(k)} \quad \text{und} \quad \bar{k}_i(k) = \lambda(k) \cdot e_i \cdot \bar{t}_i(k) \quad (4.29)$$

3. Schritt

```

if  $\bar{k}_i(k) > M_i$ 
    then  $z_i := 0$ 
         $BZ_j := BZ_j + p_{ji} \frac{1}{\mu_i} \quad \forall j=1, \dots, N$ 
if  $\bar{k}_j(k) \leq M_j \quad \forall j=1, \dots, N$ 
    then  $k := k+1$ 
goto 2.Schritt
    
```

Zur Verdeutlichung der MVABIO wird das numerische Beispiel von Abschnitt 4.1 hier noch einmal durchgeführt.

Beispiel

Für das in Abschnitt 4.1 gegebene Netz wird zunächst die klassische MVA bis zur 8. Iteration durchgeführt. Dort muß zum ersten mal eingegriffen werden, denn die $\bar{k}_i$ -Werte lauten wie folgt:

$$\begin{aligned} \bar{k}_1(8) &= 7.184 > 7 \\ \bar{k}_2(8) &= 0.811 \end{aligned}$$

d.h. In der 1. Station befinden sich mehr Aufträge als die Kapazität zuläßt. Also muß die 8. Iteration wiederholt werden:

Die 1. Station bekommt keinen neuen Auftrag mehr:

$$\bar{t}_1(8) = \frac{1}{\mu_1} \bar{k}_1(7)$$

Die 2. Station blockiert sich, also wird ein Auftrag um die Bedienzeit der 1. Station verzögert:

$$\bar{t}_2(8) = \frac{1}{\mu_2} \left[1 + \bar{k}_2(7) \right] + \frac{1}{\mu_1}$$

Jetzt kommt man bei der 8. Iteration zu folgenden Ergebnissen:

$$\begin{aligned}\bar{k}_1(8) &= 6.191 \\ \bar{k}_2(8) &= 1.81\end{aligned}$$

Man kann also normal weiteriterieren, jetzt natürlich mit den modifizierten Formeln (Gl.4.30) für die Verweilzeiten.

Bei der 10. Iteration tritt wieder ein Kapazitätsüberlauf in der 1. Station auf:

$$\bar{k}_1(10) = 7.224 > 7$$

Nun wiederholt man die 10. Iteration, wobei man die Verweilzeit der 2. Station noch einmal um die Bedienzeit der 1. Station erhöht. Das führt zu den folgenden Ergebnissen:

$$\begin{aligned}\bar{k}_1(10) &= 6.496 \\ \bar{k}_2(10) &= 3.494\end{aligned}$$

Damit erhält man die endgültigen Ergebnisse der Leistungsgrößen wie z.B. die mittleren Verweilzeiten:

$$\begin{aligned}\bar{t}_1(10) &= \underline{13.178} \\ \bar{t}_2(10) &= \underline{7.069}\end{aligned}$$

und den Durchsatz:

$$\lambda(10) = \underline{0.494}$$

Die Abweichungen werden in der folgenden Tabelle gezeigt:

	Δ in %
$\bar{k}_1$	2.35
$\bar{k}_2$	4.09
$\bar{t}_1$	1.2
$\bar{t}_2$	5
λ	1

Die gleichen Beispiele wie in Abschnitt 4.1. bzw. 4.2. werden auch hier durchgeführt und die Ergebnisse mit denen der Simulation verglichen.

Beispiel	$\bar{k}_1$	$\bar{k}_2$	$\bar{t}_1$	$\bar{t}_2$	λ
1	10.3796	9.62	34.273	31.766	0.3028
2	17.5783	7.4217	14.228	6.0073	1.235
3	5.2236	9.7764	7.6128	14.2477	0.686
4	3.9092	4.0907	38.5678	40.358	0.1013
5	30.4126	19.587	122.372	78.8139	0.2485

Die Abweichung von den Simulationswerten für die Verweilzeiten $\bar{t}_i$, den Durchsatz λ und die mittlere Anzahl von Aufträgen $\bar{k}_i$ veranschaulicht die nachstehende Tabelle.

Beispiel Nr.	Δ in %				
	$\bar{t}_1$	$\bar{t}_2$	$\bar{k}_1$	$\bar{k}_2$	λ
1	6.31	6.47	5.81	7.04	0.50
2	2.74	5.89	2.75	5.86	--
3	22.3	7.39	18.0	13.3	5.18
4	3.14	2.37	0.41	0.39	2.73
5	1.61	0.44	0.52	0.82	1.13

Die Tafel weist einige Abweichungen vom Simulationswert auf. Sie treten dann auf, wenn die Bedienraten relativ nahe beieinander liegen. Da dies der Fall ist, tritt der Blockierfall für eine der beiden Bedienstationen relativ spät ein, sofern die Kapazität eine bestimmte Größe hat. Dadurch werden fast dieselben Werte erzielt, wie sie sich auch unter normalen Bedingungen, also ohne Blockierung, ergeben würden.

BEWERTUNG

MVABLO wurde auf der Rechenanlage CDC Cyber 173 in der Programmiersprache PASCAL implementiert. Es wurden ca. 75 Beispiele durchgeführt und die Ergebnisse mit denen der Simulation verglichen. Die Ergebnisse von MVABLO weisen im Durchschnitt eine Abweichung von etwa 12 % auf. Diese Abweichungen rühren von der sehr sprunghaften Arbeitsweise von MVABLO her, die durch die Ja/Nein - Abfrage, ob die durchschnittliche Anzahl der Aufträge in den Stationen deren Kapazität überschritten hat, herrührt. Wird diese Abfrage knapp mit <Ja> beantwortet, so werden die Werte nachgebessert; wird sie knapp verneint, so passiert nichts. Das Verhalten unterscheidet sich dann trotz numerisch kleiner Unterschiede erheblich. Auf diese Weise erhalten wir eine Art Treppenkurve, die sich um den tatsächlichen Wert herumbewegt. Allerdings sind derartig geringe Abweichungen bei Blockiernetzen in jedem Fall als gering zu bezeichnen, zumal diese guten Ergebnisse von MVABLO mit sehr geringen Laufzeiten ermittelt werden.

Die größeren Abweichungen entstehen, wenn die mittlere Anzahl der Aufträge in einer Station durch eine Aufstockung von deren Blockierzeit die Kapazität überschreitet. Wird in diesem Fall die Kapazität relativ deutlich überschritten, so müssen wir den Durchsatz des Systems zu weit herabsenken, um die mittlere Auftragszahl der Station unter die Kapazität zu drücken. Ich bin der Überzeugung, daß dieser Fehler durch eine leichte Modifikation von MVABLO sehr gemindert werden kann.

Sehr wichtig ist die Tatsache, daß die MVABLO-Werte tendenziell richtig sind, d.h. auch bei Abweichungen über 10% sind die Werte als grobe Schätzungen dafür, welche Stationen gut gefüllt sind und welche nicht, sehr gut geeignet.

4.4. PRODUKTFORMLÖSUNGEN FÜR GESCHLOSSENE WARTESCHLANGENNETZE MIT BLOCKIERUNG

Wie schon zu Beginn des Kapitels 4 erwähnt, haben /HORD 81/ für geschlossene Warteschlangennetze mit Blockierung eine Produktformlösung angegeben. In diesem Abschnitt möchte ich zunächst das Theorem vorstellen und danach drei verschiedene Algorithmen für die Bestimmung der Normalisierungskonstanten und neue Formeln für die Leistungsgrößen geben.

Wir betrachten geschlossene Warteschlangennetze, die alle zum Beginn des Kapitels 4 gemachten Annahmen erfüllen.

Sei k_i die Anzahl der Aufträge in der i -ten Station (wartend + in Bedienung) und der Zustand des Netzwerks ist definiert durch den folgenden Vektor:

$$\underline{k} = (k_1, \dots, k_N)$$

Unter der Annahme, daß die Matrix der Übergangswahrscheinlichkeiten $P = [p_{ij}]$ irreduzibel sei,

d.h. $(\forall i, j \exists i_1, \dots, i_n \text{ so daß } p_{i_1 i_1} p_{i_1 i_2} \dots p_{i_{n-1} i_n} p_{i_n j} > 0)$

existiert eine Lösung zu dem linearen Gleichungssystem, (Gl. A1.2)

Sei $p(k_1, \dots, k_N)$ die Wahrscheinlichkeit, daß das Netz sich im Zustand $\underline{k}$ befindet. Diese Wahrscheinlichkeiten aller Systemzustände im statistischen Gleichgewicht zu bestimmen, ist einer der Hauptprobleme der analytischen Modellbildung. Denn aus diesen Zustandswahrscheinlichkeiten lassen sich wie schon in den vorhergehenden Abschnitten alle wichtigen Leistungsgrößen leicht ableiten.

Das folgende Theorem von /HORD 81/ liefert die Lösung für die Zustandswahrscheinlichkeiten in geschlossenen Netzen mit endlichen Warteschlangenkapazitäten.

THEOREM

Die Wahrscheinlichkeit für den Gleichgewichtszustand $\underline{k}$ in einem geschlossenen Netzwerk mit endlichen Warteschlangenkapazitäten ergibt sich durch das Produkt der Randwahrscheinlichkeiten der einzelnen Stationen:

$$p(k_1, \dots, k_N) = \frac{1}{G(K)} \prod_{i=1}^N x_i^{k_i} \delta_i(k_i) \quad (4.31)$$

wobei $G(K)$ die Normalisierungskonstante ist, so daß die Summe aller Wahrscheinlichkeiten auf 1 normiert wird.

x_i ermittelt man aus der Gl. A1.3

Die binäre Funktion $\delta_i(k_i)$ ergibt sich aus

$$\delta_i(k_i) = \begin{cases} 0 & k_i \geq M_i \\ \text{falls} & \\ 1 & k_i < M_i \end{cases}$$

Durch $\delta_i(k_i)$ werden alle Flußwahrscheinlichkeiten in oder aus nicht ausführbaren Zuständen (aufgrund der Kapazitäten) gleich 0 gesetzt. M.a.W. Durch diese Funktion werden die Zustände eliminiert, die die Kapazitäten der einzelnen Stationen überschreiten. Der Beweis kann in /HORD 81/ nachgelesen werden.

Wie in Gl.(4.31) ersichtlich, muß die Normalisierungskonstante berechnet werden, um die Zustandswahrscheinlichkeiten zu erhalten. In diesem Abschnitt werde ich Algorithmen für die Berechnung der Normalisierungskonstanten für Netze mit beschränkten Kapazitäten vorstellen.

1. Normalisierungskonstante

a) Faltungsalgorithmus für Blockiernetze

Die Normalisierungskonstante ergibt sich aus der Faltung von N Vektoren

$$G = G_1 \otimes G_2 \otimes \dots \otimes G_N \quad (4.32)$$

wobei G_i (für $i=1, \dots, N$) ein $(K+1)$ -dimensionaler Vektor ist mit

$$G_i = \begin{bmatrix} x_i(0) \\ x_i(1) \\ \cdot \\ \cdot \\ x_i(K) \end{bmatrix}$$

Die Komponenten $x_i(\cdot)$ lassen sich wie folgt bestimmen:

$$x_i(k) = \begin{cases} 1 & k=0 \\ x_i^k & k=1, \dots, M_i \\ 0 & k=M_i+1, \dots, K \end{cases} \quad \begin{matrix} \text{für } i=1, \dots, N \end{matrix} \quad (4.33)$$

Falls die Anzahl der Aufträge in der i -ten Station die Kapazität M_i überschreitet, werden die Komponenten auf 0 gesetzt, damit die nichtausführbaren Zustände wegen der Endlichkeit der Stationen eliminiert werden.

b) Erweiterung des Algorithmus von /BUZE 71/ auf Netze mit Blockierung

Zunächst wird eine Hilfsfunktion $g(k,n)$ definiert mit der Eigenschaft, daß der erste Parameter die Anzahl der Aufträge und der zweite die Stationen im System zeigt.

Der Algorithmus wird wie folgt durchgeführt:

Die Initialwerte

$$g(0,n) = 1 \quad n=1, \dots, N$$

$$g(k,1) = \begin{cases} x_1^k & k=1, \dots, M_1 \\ 0 & k=M_1+1, \dots, K \end{cases} \quad \text{für} \quad (4.34)$$

Die Hilfsfunktion $g(k,n)$ ergibt sich aus:

$$g(k,n) = g(k,n-1) + x_n g(k-1,n) \quad (4.35)$$

$$\text{für } n=2, \dots, N \quad \wedge \quad k=1, \dots, M_n$$

Wenn die Kapazität der Station M überschritten wird, d.h. für $k > M_n$, läßt sich $g(k,n)$ wie folgt ermitteln:

$$g(k,n) = g(k,n-1) + x_n g(k-1,n) - h(1,n) \quad (4.36)$$

Die Funktion $h(1,n)$ eliminiert die nicht ausführbaren Zustände. Sie ergibt sich aus:

$$h(1,n) = \frac{1}{1} \sum_{i=1}^{n-1} \sum_{j=1}^1 x_1^j h(1-j,n) \quad \text{für } l=0,1,\dots,K \quad (4.37)$$

mit dem Initialwert $h(0,n)=1$ für $n=1,\dots,N$.

Das Ziel des Algorithmus ist die Bestimmung des Wertes der Funktion $g(K,N)$, weil dieser Wert die Normalisierungskonstante $G(K)$ ist.

c) LBANCBLO

(Local Balance Algorithm for Normalizing Constant in Networks with Blocking)

Ausgehend von der MVA entwickelten /SAUE 81/ einen Algorithmus mit der Bezeichnung LBANC, der aber nur auf geschlossene Produktformnetze anwendbar ist. Für die geschlossenen Netzwerke mit endlichen Kapazitäten kann der Algorithmus in der gegebenen Form nicht angewandt werden. Daher stelle ich im folgenden eine modifizierte Version von LBANC vor.

LBANCBLO benutzt die folgende Rekursionsformel, um die unnormalisierte mittlere Warteschlangenlänge zu bestimmen:

$$q_i(k) = x_i \left[G(k-1) + q_i(k-1) - x_i^{M_i} * (M_i+1) * h_i(1) \right] \quad (4.38)$$

mit den Initialwerten

$$G(0) = 1 \quad q_i(0) = 0 \quad \text{für } i=1, \dots, N$$

$$h_i(1) = \begin{cases} \frac{1}{1} \sum_{\substack{j=1 \\ j \neq i}}^N s_j(1) & \text{für } 1 \leq M_i \\ 0 & \text{für } 1 \leq M_i \end{cases}$$

$$s_j(1) = x_j \left[h_i(1-1) + s_j(1-1) \right] \quad \text{für } j=1, 2, \dots, N$$

$$\text{mit den Initialwerten } s_j(0) = 0 \quad h_i(0) = 1$$

Die Normalisierungskonstante wird dann wie folgt ermittelt:

$$G(K) = \frac{1}{K} \sum_{i=1}^N q_i(K) \quad (4.40)$$

Mit Hilfe der Normalisierungskonstanten können alle Leistungsgrößen ermittelt werden.

2. Leistungsgrößen

Da wir Netze mit Blockierungen behandeln, können wir die klassischen Formeln /BOAK 82/ für die Leistungsgrößen nicht verwenden. Im folgenden werde ich Formeln für die Leistungsgrößen einführen, die über den ausführbaren (feasible) Zustandsraum gelten.

Die Randwahrscheinlichkeiten, d.h. die Wahrscheinlichkeiten dafür, daß in der i-ten Station genau k Aufträge sind, ergibt sich aus

$$p_i(l) = \frac{x_i^l}{G(K)} G_i^{-(K-l)} \quad (4.41)$$

wobei G_i die Normalisierungskonstante ist, die ohne Bedienstation i berechnet wird.

Die mittlere Anzahl von Aufträgen in der i-ten Station ergibt sich aus:

$$\bar{k}_i = \frac{M_i}{\sum_{l=1}^M l} \cdot p_i(l) \quad (4.42)$$

Durch Einsetzen der Gl.(4.41) erhält man

$$\bar{k}_i = \frac{1}{G(K)} \sum_{l=1}^{M_i} l \cdot x_i^l G_i^{-(K-l)} \quad (4.43)$$

Den Durchsatz des Systems bestimmt man aus

$$\lambda(K) = K \left[\sum_{i=1}^N x_i + \frac{\sum_{i=1}^N \sum_{l=1}^{M_i} l x_i^l G_i^{-(K-l-1)}}{G(K-1)} \right]^{-1} \quad (4.44)$$

Der Durchsatz der i-ten Station ergibt sich dann aus der Gl.(4.21).

Die mittlere Verweilzeit der Aufträge in der i-ten Station kann durch die Anwendung von Littles Gesetz (Gl.(4.11)) bestimmt werden. Die anderen Leistungsgrößen bestimmt man durch die Anwendung der Formeln, die in Abschnitt 4.1 (Gln.(4.12)-(4.15)) gegeben wurden.

Zur Verdeutlichung der Algorithmen und Formeln für die Leistungsgrößen wird das gleiche Beispiel wie in Abschnitt 4.1 durchgeführt.

Beispiel

Aus Gl.(4.32) kann die Normalisierungskonstante wie folgt ermittelt werden:

$$G = G_1 \otimes G_2$$

wobei

$$G_1 = \begin{bmatrix} x_1(0) = 1 \\ x_1(1) = 2 \\ x_1(2) = 4 \\ x_1(3) = 8 \\ x_1(4) = 16 \\ x_1(5) = 32 \\ x_1(6) = 64 \\ x_1(7) = 128 \\ x_1(8) = 0 \\ x_1(9) = 0 \\ x_1(10) = 0 \end{bmatrix} \quad G_2 = \begin{bmatrix} x_2(0) = 1 \\ x_2(1) = 0.9 \\ x_2(2) = 0.81 \\ x_2(3) = 0.729 \\ x_2(4) = 0.656 \\ x_2(5) = 0.59 \\ x_2(6) = 0.53 \\ x_2(7) = 0.478 \\ x_2(8) = 0.43 \\ x_2(9) = 0 \\ x_2(10) = 0 \end{bmatrix}$$

Damit erhält man

$$G = \begin{bmatrix} 1 \\ 2.9 \\ 6.61 \\ 13.949 \\ 28.55 \\ 57.698 \\ 115.92 \\ 232.33 \\ 209.09 \\ 187.783 \\ 168.216 \end{bmatrix}$$

Jetzt sind wir in der Lage, durch Verwendung der gegebenen Formeln die Leistungsgrößen für dieses Netz zu ermitteln:

Die mittlere Anzahl von Aufträgen (Gl.(4.43))

$$\bar{k}_1 = \frac{1}{G(K)} \sum_{l=1}^7 l \cdot x_1^l \cdot G_2(K-1) = \underline{6.2322}$$

$$\bar{k}_2 = \frac{1}{G(K)} \sum_{l=1}^8 l \cdot x_2^l \cdot G_1(K-1) = \underline{3.7672}$$

Der Durchsatz (Gl.(4.44))

$$\lambda_1 = \underline{0.499}$$

$$\lambda_2 = \underline{0.499}$$

Die mittlere Verweilzeit (Gl.(4.11))

$$\bar{t}_1 = \frac{\bar{k}_1}{\lambda_1} = \underline{12.489}$$

$$\bar{t}_2 = \underline{7.549}$$

Die Abweichungen fasse ich in der folgenden Tabelle zusammen:

	Δ in %
$\bar{k}_1$	6.68
$\bar{k}_2$	11.
$\bar{t}_1$	6.8
$\bar{t}_2$	10.9
λ	0

Wie man leicht aus der Tabelle feststellen kann, liefert diese Vorgehensweise schlechtere Ergebnisse als die MVABLO (Abschnitt 4.3) und die Zustandsraumtransformationsmethode (Abschnitt 4.1. bzw. 4.2).

Bewertung

Zur Bewertung wurden die in Abschnitt 4.4 vorgestellten Algorithmen auf der Rechenanlage CDC-Cyber 173 am RRZE in Erlangen implementiert. Es wurden insgesamt mehr als 75 Netzwerke zur Bewertung getestet.

Da die Produktformvorgehensweise die Kopplung der Stationen im Netz nicht berücksichtigt und damit die Effekte einer Blockierung nur unzureichend nachvollzieht, sind die Abweichungen der Ergebnisse von denen der Simulation sehr stark von den Eigenschaften des betrachteten Netzes abhängig.

Netze, die keine ausgeprägte Engpaßstation besitzen und solche, in denen aufgrund von low traffic nur relativ wenige Blockierungen vorkommen, werden von der Produktformvorgehensweise mit gutem Ergebnis analysiert. Hier liegen die Abweichungen für die mittleren Auftragszahlen bei 10-12% und für den Durchsatz bei 0-10%. Höhere Abweichungen ergeben sich nur bei numerisch kleinen Werten.

Doch bei Netzen, in denen ausgeprägte Engpässe und demzufolge viele Blockierungen vorkommen, können sehr große Abweichungen auftreten. In diesen Fällen ist aus dem Ergebnis nicht einmal die ungefähre Verteilung der Aufträge im Netz zu erkennen, d.h. die Ergebnisse sind auch zu groben Schätzungen nicht verwendbar.

Generell lagen die Ergebnisse von MVABLO (Abschnitt 4.3.) fast immer besser als die der Produktformvorgehensweise. Somit hat sich in der Validierung die oben erwähnte Tatsache, daß Blockiernetze keine Produktformlösungen haben, bestätigt.

Die gleichen Beispiele wie in Abschnitt 4.1, 4.2 bzw. 4.3 werden auch hier mit den oben eingeführten Produktformalgorithmen durchgeführt und die Ergebnisse mit denen der Simulation verglichen.

Beispiel	$\bar{k}_1$	$\bar{k}_2$	$\bar{t}_1$	$\bar{t}_2$	λ
1	11.41	8.59	31.23	23.52	0.365
2	16.60	8.39	12.12	6.13	1.369
3	6.67	8.37	9.48	11.84	0.703
4	3.68	4.32	26.48	31.05	0.139
5	30.01	19.90	98.78	65.33	0.304

Die Abweichungen von den Simulationswerten werden in der folgenden Tabelle gezeigt:

Beispiel	Δ in %				
	$\bar{k}_1$	$\bar{k}_2$	$\bar{t}_1$	$\bar{t}_2$	λ
1	3.418	4.62	17.1	16.8	17.5
2	3.1	6.09	14.2	4.1	17.1
3	4.5	3.11	3.38	12.0	7.5
4	6.657	5.67	50.3	33.1	29.0
5	1.56	2.38	25.9	21.1	24.0

Hier stellt man auch fest, daß die Ergebnisse deutlich schlechter sind als bei der MVABLO und der Zustandsraumtransformationmethode.

5. ANALYSE DES GESAMTMODELLS

Im Abschnitt 3 wurde ein Gesamtmodell für das Multiprozessor-system (Fig. 3.2) entwickelt. Dieses Modell kann durch die folgenden Annahmen präzisiert werden.

- Das Modell besteht aus N Bedienstationen, wobei die erste Station die Prozessoren modellieren. Die Stationen 2 bis N enthalten so viele Bedieneinheiten, daß keine Abarbeitungsstrategie für Prozesse notwendig ist. Diese Bedienstationen bilden den Blockierzustand ab, wobei die mittleren Bedienzeiten den Blockierzeiten der Prozesse entsprechen.
- Es zirkulieren insgesamt K-Prozesse im System.
- Die Bedienzeit eines Prozesses in der i-ten Station ist exponentiell verteilt mit dem Mittelwert $1/\mu_i$ ($i=1,...,N$).
- Die Abarbeitungsstrategie in der ersten Station ist PS (Processor Sharing), d.h. allen Prozessen wird eine infinitesimal kleine Zeitscheibe für die Bedienung zugeteilt. Dies erweckt den Eindruck, daß alle Prozesse aktiv wären.

Wie schon im Abschnitt 3.3 erwähnt, müssen die Blockierzeiten der Prozesse, die zu den mittleren Bedienzeiten der Verzögerungsstationen und die Blockierwahrscheinlichkeiten der Prozesse, die den Übergangswahrscheinlichkeiten von den Prozessoren zu den Verzögerungsstationen entsprechen, berechnet werden. Für die Bestimmung dieser Größen wurde ein Software-Modell im Abschnitt 3.2 entwickelt und neue Methoden dafür im Kapitel 4 eingeführt. Damit können die aus dem Software-Modell gewonnenen Werte für Blockierzeiten und Blockierwahrscheinlichkeiten als Eingabeparameter für die Analyse des Gesamtmodells eingesetzt werden.

Die Übergangswahrscheinlichkeiten vom Multiprozessor zu den einzelnen Verzögerungsstationen ergeben sich aus den Quotienten der jeweiligen Blockierwahrscheinlichkeit und der Anzahl der Aufträge im System. Auf diese Weise werden die Wahrscheinlichkeiten p_{1j} ($j=2,...,N$) bestimmt. (vgl. Gl. 3.2).

Da ein Prozeß nach dem Verlassen einer Verzögerungsstation auf jeden Fall wieder zum Multiprozessor zurückkehrt, betragen die Übergangswahrscheinlichkeiten p_{j1} ($j=2,...,N$) alle 1; Übergänge zwischen den Verzögerungsstationen sind nicht möglich, so daß die

übrigen Wahrscheinlichkeiten p_{ij} ($i=2,\dots,N$; $j=2,\dots,N$) alle gleich 0 sind. Damit sind die Übergangswahrscheinlichkeiten für alle möglichen Übergänge zwischen den Stationen des Warteschlangenmodells angegeben.

Da die zur Bearbeitung anstehenden Prozesse in den Prozessoren nach Warteschlangenstrategie PS abgearbeitet werden, stellt sich die Situation bei sehr kleinen Zeitscheiben nach außen hin so dar, als ob alle Prozesse simultan bearbeitet würden und zwar unabhängig von der Anzahl der Prozessoren.

Im Systemmodell, Kap.2, wurde angenommen, daß jeder Prozeß mit jedem kommunizieren kann, wenn er aktiv ist, d.h. auf dem Prozessor läuft. In dem entwickelten Software-Modell, Abschnitt 3.2, können die Prozesse Nachrichten austauschen, als ob alle aktiv wären. Da bei der PS alle Prozesse bei sehr kleiner Zeitscheibe simultan aktiv sein können, ist unsere Überlegung doch richtig, daß in diesem Fall die Blockierwahrscheinlichkeiten und -zeiten aus dem Software-Modell zu gewinnen. Dies zeigen auch die folgenden Ergebnisse, die analytisch schon im Kapitel 4 erzielt wurden. Die Vergleichswerte wurden aus der Simulation des Gesamtmodells, Anhang 2, erhalten.

Beispiel 1

$$K = 20; M_1 = 18; M_2 = 12; 1/\mu_1 = 2; 1/\mu_2 = 3.3333;$$

	Analytisch	Simulation	Abweichung in %
P_{B_1}	0.4015	0.4082	1.6414
P_{B_2}	0.0024	0.0040	40.0
$\bar{t}_{B_1}$	31.286	30.851	1.409
$\bar{t}_{B_2}$	-	-	-

Die hohe Abweichung der zweiten Blockierwahrscheinlichkeit ergibt sich aus den sehr geringen absoluten Werten; insbesondere zeigt hier die Simulation abweichende Ergebnisse, weil zum einen

Teil bei solch kleinen Werten die Einschwingphase ein überproportional hohes Gewicht erhält und sich andererseits Rundungsfehler bei der Berechnung der Blockierwahrscheinlichkeiten unverhältnismäßig stark bemerkbar machen. Die sehr geringe Differenz in den absoluten Werten zeigt jedoch, daß auch diese Werte grundsätzlich sehr dicht nebeneinander liegen.

Beispiel 2

$$K = 25; M_1 = 18; M_2 = 13; 1/\mu_1 = 0.8; 1/\mu_2 = 0.5;$$

	Analytisch	Simulation	Abweichung in %
P_{B_1}	0.0143	0.0087	39.161
P_{B_2}	0.3842	0.3805	0.9630
$\bar{t}_{B_1}$	-	-	-
$\bar{t}_{B_2}$	3.047	2.996	1.697

Auch hier ist wieder die Übereinstimmung im größeren und die Abweichung im sehr kleinen Wert zu erkennen; in absoluten Zahlen sind die zueinandergehörigen Größen jedoch wiederum als nahezu identisch anzusehen.

Beispiel 3

$$K = 15; M_i = 10 \text{ für } i=1,2; 1/\mu_1 = 1.1111; 1/\mu_2 = 1.5;$$

	Analytisch	Simulation	Abweichung in %
P_{B_1}	0.2851	0.2975	4.1681
P_{B_2}	0.0349	0.0476	26.681
$\bar{t}_{B_1}$	5.7077	5.429	5.13
$\bar{t}_{B_2}$	-	-	-

Man erkennt deutlich: Je weniger extrem die Werte sind, um so geringer ist die Abweichung, was obige These unterstützt.

Beispiel 4

$K = 8$; $M_1 = 4$; $M_2 = 7$; $1/\mu_1 = 10$; $1/\mu_2 = 2.5$;

	Analytisch	Simulation	Abweichung in %
P_{B_1}	0.0007	0.0030	76.667
P_{B_2}	0.7502	0.7588	0.6525
$\bar{t}_{B_1}$	-	-	-
$\bar{t}_{B_2}$	37.51	37.145	0.972

Beispiel 5

$K = 50$; $M_1 = 31$; $M_2 = 24$; $1/\mu_1 = 4$; $1/\mu_2 = 2$;

	Analytisch	Simulation	Abweichung in %
P_{B_1}	0.0039	0.0068	42.647
P_{B_2}	0.5102	0.5101	0.0196
$\bar{t}_{B_1}$	-	-	-
$\bar{t}_{B_2}$	74.1732	73.711	0.627

Im folgenden werde ich 3 verschiedene Multiprozessorsysteme mit unterschiedlichen Kombinationen durchführen und die Ergebnisse mit der Simulation vergleichen. Als Leistungsgrößen werden die Auslastung, der Durchsatz, die mittlere Anzahl von Aufträgen, des Multiprozessors und die mittlere Verweilzeit der Prozesse berechnet.

5.1. BEISPIEL 1

Hier wird der im Abschnitt 3.2 in der Fig.3.4 gezeigte Spezialfall, daß im Multiprozessorsystem jeweils 2 Prozesse miteinander kommunizieren, analysiert. Das Systemmodell, wie in Kap.2, wie folgt beschrieben:

Es gibt $m_1 = 1, 4, 8, 16, 20$ Prozessoren im System. Die Bedienzeit der Prozessoren ist exponentiell verteilt mit dem Mittelwert $1/\mu_1 = 61$ sec. Die Abarbeitungsstrategie der Prozessoren ist PS. Es gibt insgesamt $K=20$ Prozesse im System. Jeder Prozeß kommuniziert nur mit einem anderen Prozeß, d.h. es bilden sich damit 10 Prozeßpaare für die Kommunikation. Die Puffer für Nachrichtenaustausch der Prozesse sind endlich und haben die Kapazitäten $M_i = 10$ für $i=1,...,20$. Pro Prozeßpaar werden insgesamt $C = 11$ Nachrichten ausgetauscht.

Die Prozesse haben im Mittel unterschiedliche Kommunikationszeiten. Die Mittelwerte werden in der folgenden Tabelle gezeigt.

Paar	$1/\mu_i$	$1/\mu_j$
1	9.69	9.26
2	10.3	9.95
3	9.91	9.9
4	9.92	9.85
5	10.34	9.26
6	10.79	9.06
7	10.78	10.11
8	9.19	9.72
9	9.52	11.08
10	11.17	9.13

Gemäß dem Modellierungsprozeß, Abschnitt 1.2, muß ein Warteschlangennetz entwickelt werden, um das oben beschriebene Multiprozessorsystem zu analysieren. Die Analyse erfolgt nach unserem Konzept hierarchisch.

SOFTWARE-MODELL

Zunächst entwickeln wir das Software-Modell, das aus 10 völlig unabhängig voneinander kommunizierenden Prozessen besteht.

Durch die Analyse dieses Software-Modells mit der Zustands-raumtransformationmethode, Kap. 4.1, erhalten wir die folgenden Blockierwahrscheinlichkeiten und Blockierzeiten für jeden Prozeß wie folgt:

Prozeß Paar	P_{B_i}			$\bar{t}_{B_i}$		
	An.	Sim.	Abw.	An.	Sim.	Abw.
1	0	0	-	0	0	-
2	0.491	0.489	0	0.431	0.428	0
3	0.005	0	-	0	0	-
4	0.472	0.485	2.7	0.348	0.354	1.7
5	0.032	0	-	0	0	-
6	0.041	0.036	14.0	0	0.011	-
7	0.009	0	-	0	0	-
8	0.357	0.430	17.0	0.068	0.074	7.4
9	0.014	0	-	0	0	-
10	0.476	0.495	3.8	1.082	1.078	0
11	0.013	0	-	0	0	-
12	0.375	0.497	24.5	1.729	1.734	0.3
13	0.009	0	-	0	0	-
14	0.432	0.492	12.2	0.665	0.677	1.8
15	0.424	0.491	13.6	0.527	0.527	0
16	0.006	0	-	0	0	-
17	0.466	0.496	6.0	1.551	1.558	0
18	0.046	0	-	0	0	-
19	0.024	0	-	0	0	-
20	0.554	0.497	11.5	2.040	2.043	0

Der Vergleich zeigt, daß die Abweichungen sehr gering sind.

GESAMTMODELL

Das Gesamtmodell besteht aus $N=21$ Stationen, wobei die 1. Station die Prozessoren als $M/M/m_1$ -PS und die Stationen 2-20 die Verzögerungen der Prozesse als $M/M/IS$ -modellieren.

Jeder Prozeß kehrt nach dem Verlassen einer Verzögerungsstation zur 1. Station mit der Wahrscheinlichkeit $p_{j1} = 1$ ($j=2, \dots, 20$) zurück. Um die Übergangswahrscheinlichkeiten der Prozesse von der 1. Station zur 2. Station zu bestimmen, werden wir die aus dem Software-Modell ermittelten Blockierwahrscheinlichkeiten durch die Anwendung der Formeln Gl. 3.1 bzw. 3.2 umrechnen. Damit haben wir ein komplettes Modell zu analysieren. Für die Analyse benutzen wir den TUPAL-Algorithmus.

Die analytischen Ergebnisse sind in der folgenden Tabelle zusammen mit den Simulationswerten und der prozentualen Abweichung der entsprechenden Größen angegeben. Sie werden nachfolgend durch Graphiken verglichen. In den Figuren gibt die durchgezogene Linie die Werte aus der Simulation und die gestrichelte Linie die analytischen Ergebnisse wieder. Die x-Achse zeigt verschiedene Anzahlen von Prozessoren.

	Analytisch		Simulation	Abweichung
$m_1=1$	ρ	1	1	0
	λ	0.01	0.01	9.75
	$\bar{k}$	17.68	19.22	8.01
	$\bar{t}$	1077.18	1218.24	11.57
$m_1=4$	ρ	1	1	0
	λ	0.06	0.05	10.19
	$\bar{k}$	17.01	19.22	11.49
	$\bar{t}$	259.04	304.56	14.91
$m_1=8$	ρ	1	1	0
	λ	0.13	0.11	9.91
	$\bar{k}$	18.36	19.22	4.47
	$\bar{t}$	139.79	152.28	8.21
$m_1=16$	ρ	0.999	1	0
	λ	0.262	0.23	9.93
	$\bar{k}$	19.421	19.22	1.03
	$\bar{t}$	73.951	76.14	2.87
$m_1=20$	ρ	0.96	0.9609	0
	λ	0.31	0.3155	0
	$\bar{k}$	19.22	19.22	0
	$\bar{t}$	60.91	60.94	0

Auslastung

Die analytischen und simulativen Ergebnisse für die Auslastung des Multiprozessorsystems werden in der folgenden Fig. 5.3 gegenübergestellt:

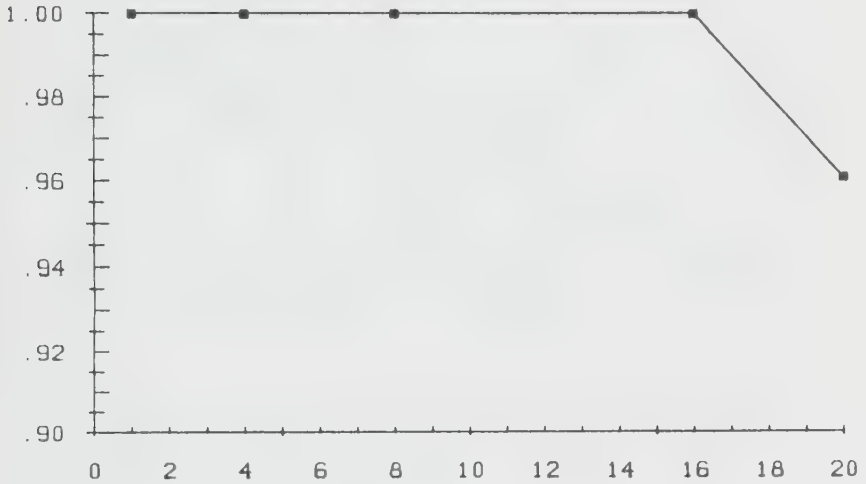


Fig. 5.3

Bemerkenswert ist die Tatsache, daß keinerlei Abweichung zu beobachten ist. Die größte Abweichung zwischen Simulations- und analytischen Werten beträgt 0.0056.

Durchsatz

In Fig. 5.4 werden die analytischen und simulativen Werte für den Durchsatz des Multiprozessors verglichen:

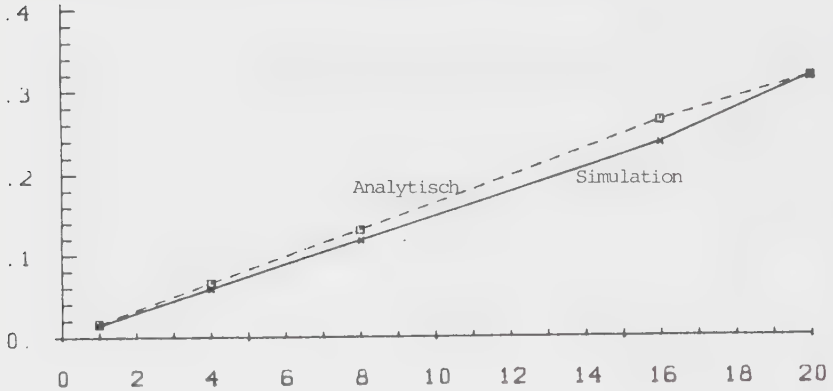


Fig.5.4

Die Abweichungen bewegen sich zumeist im Durchschnitt um 9%, wobei bei 20 Prozessoren die Werte völlig übereinstimmen. Der Grund liegt darin, daß alle Prozesse (im System befinden sich ja $K=20$ Prozesse) aktiv sind, d.h. es bildet sich keine Ablaufwarteschlange.

Mittlere Verweilzeit der Prozesse

In der folgenden Fig. 5.5 werden die Ergebnisse für die mittlere Verweilzeit der Prozesse auf den Prozessoren vorgestellt:

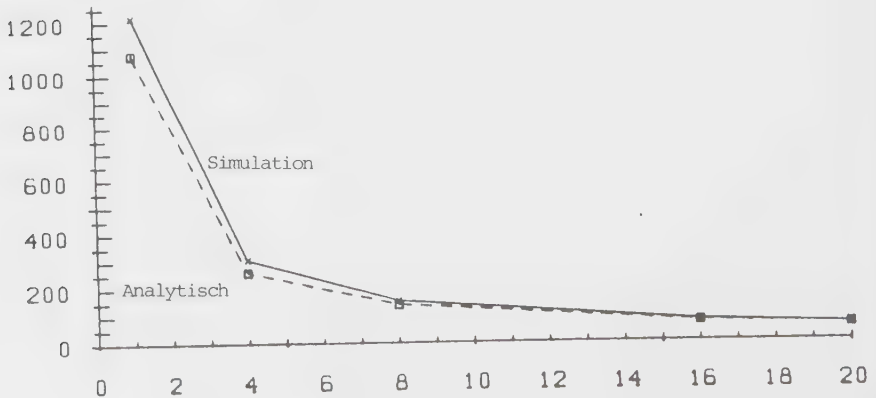


Fig. 5.5

Außer im Bereich von 1 bis 12 Prozessoren liegen die analytischen und Simulationsergebnisse nahe beisammen. Die höchste Abweichung beträgt 14.9% bei 4 Prozessoren. Wie beim Durchsatz gilt auch hier, daß bei 20 Prozessoren keine Abweichungen vorkommen.

Mittlere Anzahl von Aufträgen

Die Simulationsergebnisse bleiben fast konstant bei jeder Anzahl von Prozessoren. Daher ist die Kurve für die Simulation eine Gerade. Die analytischen Ergebnisse weichen bei 4 Prozessoren am meisten ab ca. 11.5%, was auf der Fig. 5.6 auch deutlich sichtbar ist. Je mehr Prozessoren im System vorhanden sind, desto mehr nähern sich die analytischen Ergebnisse denen der Simulation. Bei 16 Prozessoren übersteigt der analytische Wert sogar den der Simulation, ohne sich aber weit von diesen zu entfernen. Für 20 Prozessoren stimmen die Werte völlig überein.

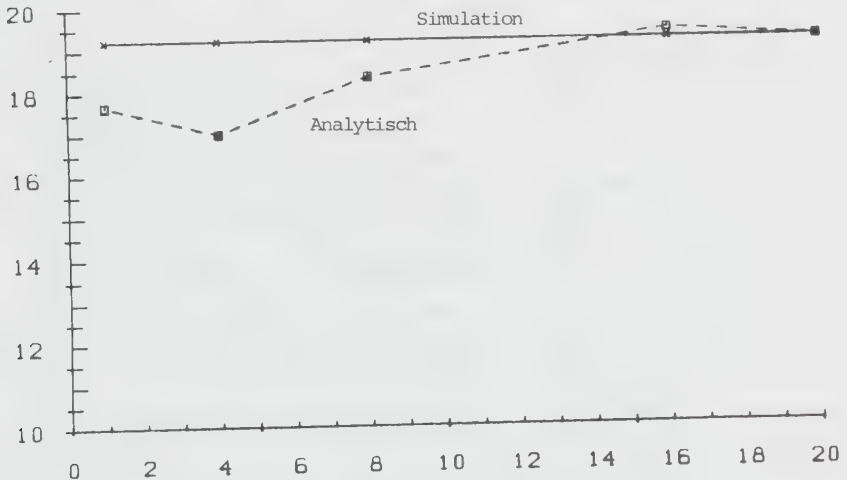


Fig. 5.6

5.2 BEISPIEL 2

Das in diesem Abschnitt zu untersuchende System besteht aus $m=1,2,4$ Prozessoren und insgesamt von 4 Prozessen. Die Bedienzeit der Prozessoren ist exponentiell verteilt mit dem Mittelwert $1/\mu_1 = 59$ sec. Die Prozessorbearbeitungsstrategie ist PS. Jeder Prozess kommuniziert mit jedem Prozess gleich wahrscheinlich. Die Puffer für die Kommunikation sind endlich $M_i = 4$ ($i=1,\dots,4$) und die Gesamtanzahl der Nachrichten, die ausgetauscht werden, ist $C=14$. Die Kommunikationszeitdauer im Mittel ist in folgender Tabelle aufgelistet:

Prozeß	$1/\mu$
1	12.92
2	5.56
3	6.18
4	9.04

SOFTWARE-MODELL

Um die Blockierzeiten und -wahrscheinlichkeiten der Prozesse zu erhalten, wird ein Software-Modell mit vier Stationen und endlichen Kapazitäten entwickelt und mit der Zustandsraumtransformationemethode, Abschnitt 4.2, analysiert.

Die folgende Tabelle zeigt die analytischen und simulativen Ergebnisse und die Abweichungen voneinander.

Prozeß	P_{B_i}			T_{B_i}		
	An.	Sim.	Abw.	A.	Sim.	Abw.
1	0.001	0	-	0	0	0
2	0.589	0.6235	5.73	3.85	4.021	4.4
3	0.342	0.332	2.92	7.456	7.295	2.15
4	0.361	0.372	3.04	9.647	9.906	2.68

Wie im vorhergehenden Abschnitt müssen auch hier die aus dem Software-Modell ermittelten Blockierwahrscheinlichkeiten zu den für die Analyse des Gesamtmodells benötigten Übergangswahrscheinlichkeiten umberechnet werden.

So erhalten wir ein Warteschlangennetz, wobei die 1. Station $m=1,2,4$ Bedieneinheiten, exponentielle Bedienzeitverteilung und PS Abarbeitungsstrategie hat. Die Stationen 2,3,4 modellieren die Verzögerung der Prozesse als M/M/IS-. Stationen.

Die analytisch und simulativ ermittelten Leistungsgrößen werden in der folgenden Tabelle gezeigt; anschließend werden sie graphisch gegenübergestellt.

		An.	Sim.	Abw.
$m_1=1$	ρ	1	1	0
	λ	0.01	0.017	0.59
	$\bar{k}$	3.72	3.17	14.7
	$\bar{t}$	219.44	235.64	7.38
$m_1=2$	ρ	0.99	1	0.16
	λ	0.03	0.03	0.57
	$\bar{k}$	3.72	3.17	14.54
	$\bar{t}$	109.85	117.57	6.56
$m_1=4$	ρ	0.76	0.79	2.67
	λ	0.05	0.05	2.59
	$\bar{k}$	3.10	3.17	2.45
	$\bar{t}$	59.91	60.09	0.36

Auslastung

Fig. 5.7 zeigt die Gegenüberstellung der Auslastung des Multiprozessors mit 4 Prozessen, die analytisch und simulativ gewonnen wurden:

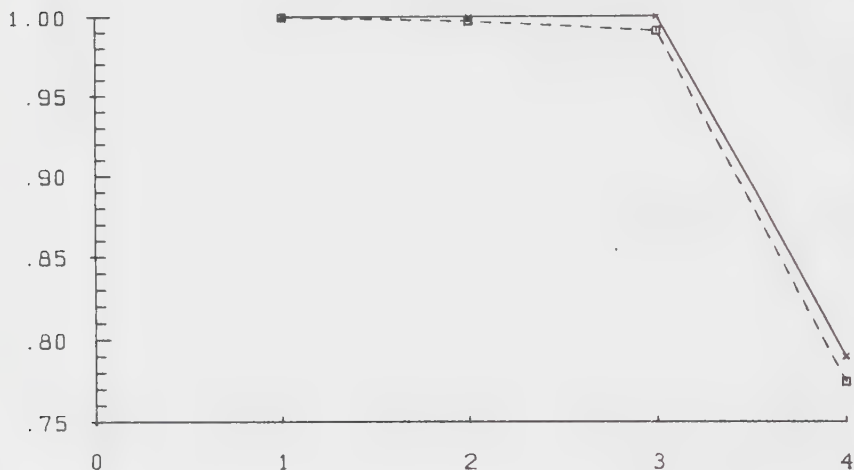
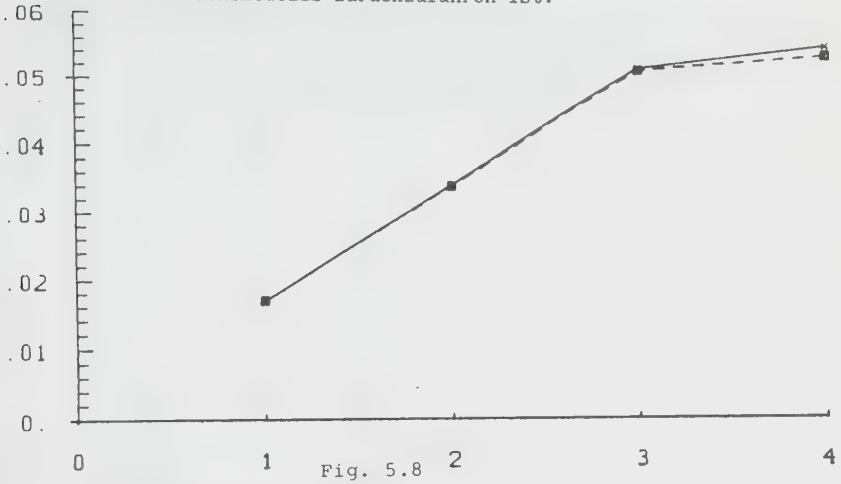


Fig. 5.7

Die Figur zeigt, daß sich die Werte bis auf geringfügige Abweichungen entsprechen. Die noch auftretenden Differenzen sind wohl im wesentlichen auf die in der Simulation auch nach 50000 Zeiteinheiten noch zu berücksichtigenden Abweichungen durch die Einschwingphase des Simulationsmodells zurückzuführen. Es ist auch logisch, daß die Werte für die Auslastungen beim 4 Prozessor Fall abnehmen, da im System nur 4 Prozesse vorhanden sind und durchaus Prozessoren manchmal Leerlaufzeiten haben können.

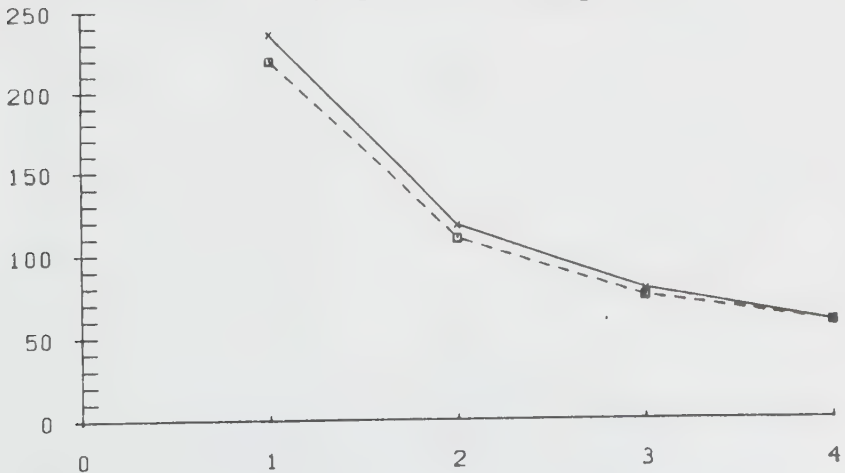
Durchsatz

Die analytischen und simulativen Werte für den Durchsatz stimmen für den Fall mit einem, zwei und drei Prozessoren völlig überein. Lediglich bei 4 Prozessoren zeigt sich eine geringfügige Abweichung, die wohl wiederum auf den Einfluß der Einschwingphase des Simulationsmodells zurückzuführen ist.



Mittlere Verweilzeit

Die graphische Gegenüberstellung zeigt eine bis auf geringfügige Abweichungen gute Übereinstimmung der Werte.



Mittlere Anzahl von Aufträgen

Wie in der Fig. 5.10 ersichtlich, bleiben die Werte der Simulation für jede Anzahl von Prozessoren konstant, wobei bei den analytischen Ergebnissen der Wert bei 20 Prozessoren abnimmt und sich der Simulation bis auf 2.5% annähert. Die Größte Abweichung in diesem Fall beträgt ca. 14.5%.

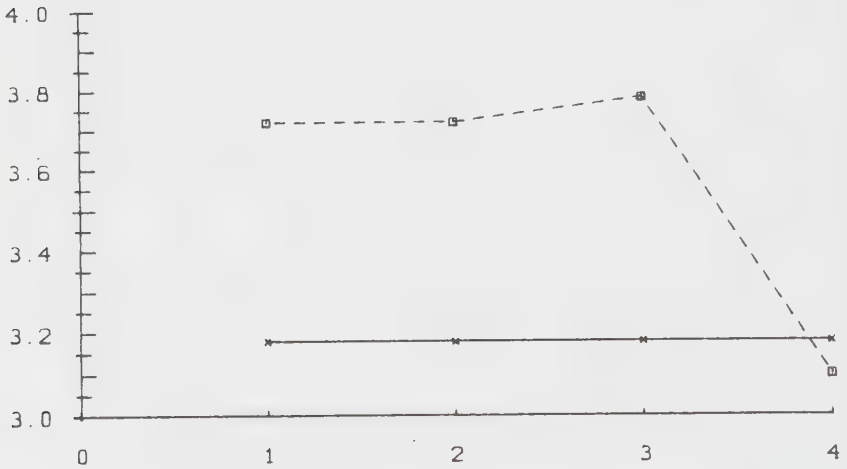


Fig. 5.10

5.3 BEISPIEL 3

Das zu analysierende System besteht aus $m = 1, 2, 4, 8, 10$ Prozessoren und $K=10$ Prozessen. Die Bedienzeit der Prozessoren ist exponentiell verteilt mit dem Mittelwert $1/\mu_i = 76\text{sec}$. Jeder Prozeß kommuniziert hier auch mit jedem gleichwahrscheinlich. Die Puffer haben endliche Kapazitäten $M_i=4, i=1,...,10$. Die Gesamtzahl der Nachrichten ist $C=35$. Die Kommunikationszeitdauern der Prozesse sind in folgender Tabelle aufgelistet.

Prozeß	$1/\mu$
1	77.52
2	5.56
3	10.31
4	9.04
5	19.82
6	9.01
7	13.89
8	13.79
9	13.72
10	13.89

SOFTWARE-MODELL

Aus dem Software-Modell werden die Blockierwahrscheinlichkeiten und -zeiten ermittelt und wie folgt mit Simulation verglichen

Prozeß Nr.	P_{B_i}			$\bar{t}_{B_i}$		
	A.	Sim.	Abw.	A.	Sim.	Abw.
1	0.465	0.4928	5.9	7.46	7.72	3.48
2	0.695	0.7133	2.6	8.23	8.50	3.2
3	0.592	0.6245	5.2	15.79	16.06	1.7
4	0.001	0.0004	-	0	4.43	-
5	0.764	0.7459	2.4	29.2	27.91	4.4
6	0.528	0.5458	3.3	9.52	10.66	11.9
7	0.179	0.1845	3.1	11.26	11.96	6.2
8	0.184	0.1995	8.4	11.98	12.35	3.1
9	0.521	0.5010	3.8	10.45	9.67	7.5
10	0.523	0.4989	4.6	13.25	12.83	3.2

GESAMTMODELL

Das Gesamtmodell wird mit TUPAL analysiert und die folgenden Ergebnisse für die Leistungsgrößen erzielt:

	Ana.	Sim.	Abw.
$m_1=1$	ρ 0.99 λ 0.01 $\bar{k}$ 8.08 $\bar{t}$ 615.81	1.0 0.01 7.19 761.18	0 0 11 23.6
$m_1=2$	ρ 0.99 λ 0.02 $\bar{k}$ 8.09 $\bar{t}$ 308.27	1.00 0.02 7.19 380.91	0 0 11.12 19
$m_1=4$	ρ 0.99 λ 0.05 $\bar{k}$ 8.77 $\bar{t}$ 167.07	1.0 0.05 7.19 190.45	0 0 22 12.27
$m_1=8$	ρ 0.82 λ 0.08 $\bar{k}$ 6.83 $\bar{t}$ 78.72	0.89 0.09 7.32 76.12	8.01 8.05 6.43 3.27
$m_1=10$	ρ 0.65 λ 0.08 $\bar{k}$ 6.61 $\bar{t}$ 76.18	0.72 0.09 7.19 75.94	8.18 8.16 8.20 0.31

Auslastung

In Fig. 5.11 ist die Auslastung des Multiprozessorsystems mit $K=10$ Prozessen und verschiedener Anzahl von Prozessoren gezeichnet.

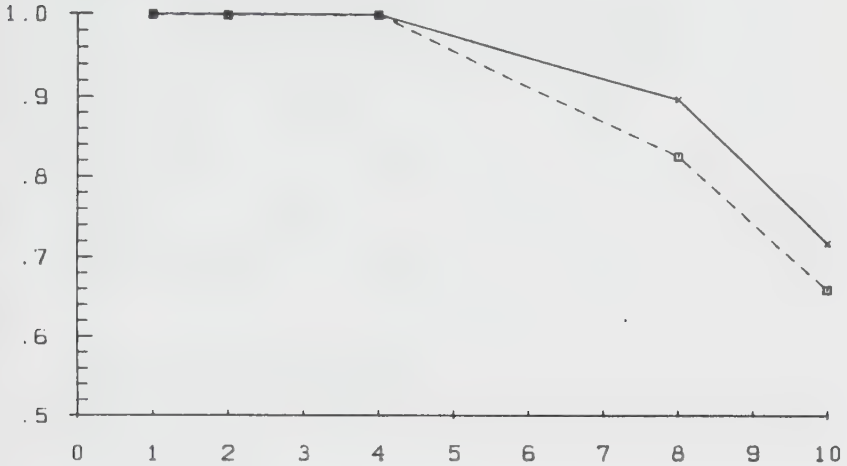


Fig.5.11

Fig. 5.11 zeigt eine nahezu exakte Übereinstimmung der analytischen und simulativen Werte für bis zu 4 Prozessoren. Bei noch mehr Prozessoren wird die Abweichung etwas höher (8%), wobei die Werte aber tendenziell immer noch übereinstimmen.

Durchsatz

In der folgenden Fig. 5.12 sind die Werte für den Durchsatz gegenübergestellt. Wie bei der Auslastung, Fig. 5.11, stimmen die Werte für bis zu 4 Prozessoren im System vollkommen überein. Die Abweichungen bei mehr als 5 Prozessoren hängen wieder damit zusammen, daß die meisten Prozessoren leer stehen, wie auch aus der Graphik für die mittlere Anzahl von Aufträgen leicht ersichtlich ist.

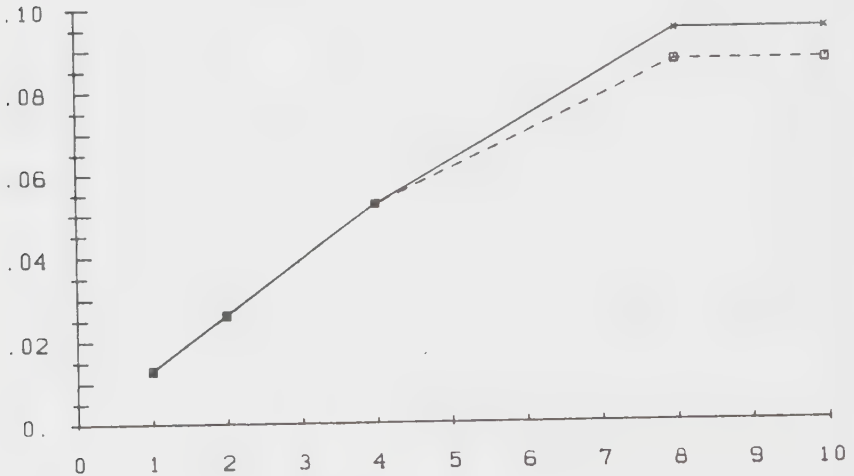


Fig. 5.12

Mittlere Verweilzeit

Je mehr Prozessoren da sind, desto mehr nimmt logischerweise die Verweilzeit der Prozesse bei den Prozessoren immer mehr ab, was der Kurvenverlauf in der Fig. 5.13 auch zeigt. Die Werte stimmen ab 8 Prozessoren vollkommen überein. In diesem Fall beträgt die höchste Abweichung bei 23%, wobei die Prozentzahl kontinuierlich abnimmt.

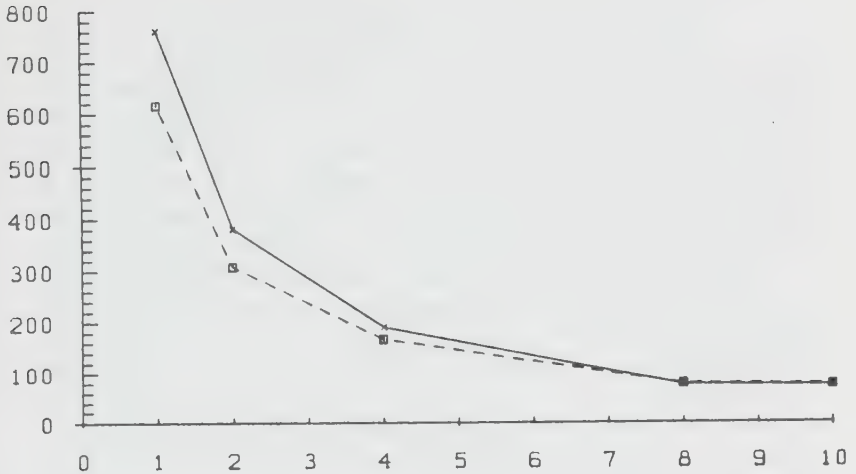


Fig. 5.13.

Mittlere Anzahl von Aufträgen

In der Fig. 5.14 ist es festzustellen, daß die Simulationswerte für jede Anzahl von Prozessoren wie bei den vorhergehenden Beispielen fast konstant bleiben. Die analytischen Werte steigen sehr geringfügig bei 4 Prozessoren, aber normalisieren sich wieder und nehmen langsam ab und nähern sich an den Simulationswerten.

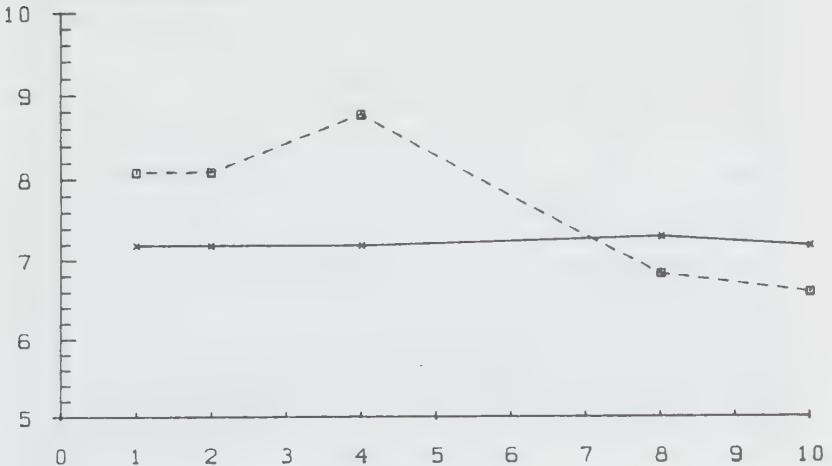


Fig. 5.14

6. ZUSAMMENFASSUNG UND VORSCHLÄGE FÜR DIE ZUKUNFT

Im Rahmen dieser Arbeit wird ein Multiprozessorsystem analysiert, in dem Prozesse über endliche Puffer Nachrichten austauschen und sich aufgrund der Kommunikation blockieren können.

Die Systemeigenschaften wurden in Kapitel 2 ausführlich beschrieben. Die Leistungsanalyse von Rechensystemen wird, wie in Abschnitt 1.2 erwähnt, so durchgeführt, daß mit Hilfe der Systembeschreibung ein entsprechendes Warteschlangennetz modelliert und dieses Modell dann mit den geeigneten Methoden analysiert wird.

Bei der Systembeschreibung, Kap. 2, hatten wir festgestellt, daß die Prozesse sich in 3 Zuständen befinden können: bereit, aktiv und blockiert. Die Prozesse gingen dann vom aktiven in den Blockierzustand über, wenn die Kommunikation (Nachrichtenaustausch) nicht erfolgreich abgelaufen war. Da aber Prozessoren, Prozesse und Nachrichten in einem einzigen Warteschlangennetz nicht berücksichtigt werden können, hatten wir in Kapitel 3 zwei Modelle (Gesamt-/Software-Modelle) entwickelt.

Im Gesamtmodell wurde der Prozeßablauf, d.h. das Zustandsdiagramm dargestellt. Die Bereit-Aktiv-Zustände wurden als Typ $M/M/m_i$ -PS-Stationen und der Blockierzustand als $M/M/IS$ -Stationen modelliert. Dabei sahen wir, daß die Übergangswahrscheinlichkeiten von der $M/M/m_i$ -PS-Station zu den $M/M/IS$ -Stationen den Blockierwahrscheinlichkeiten der einzelnen Prozesse und die mittleren Bedienzeiten der $M/M/IS$ -Stationen den mittleren Blockierzeiten der Prozesse entsprechen und als Systemparameter nicht vorgegeben werden können.

Um diese Größen zu ermitteln, hatten wir ein Software-Modell entwickelt, daß die Prozesse als abstrakte Bediener und die Nachrichten als Aufträge, die durch das System zirkulieren, beinhaltet. Die aus dem Software-Modell entwickelten Größen sollten dann in das Gesamtmodell eingesetzt und die Leistungsgrößen für verschiedene Kombinationen von Multiprozessorsystemen berechnet werden.

Die Analyse des Softwaremodells war der Schwerpunkt dieser Arbeit. Wir hatten als Software-Modell ein Warteschlangennetz mit endlichen Stationskapazitäten erhalten. Wie auch in der Literaturübersicht erwähnt, existieren für solche Netze keine eindeu-

tigen einfachen Lösungen. So mußten völlig neue Algorithmen entwickelt werden, um diese einfach untersuchen zu können.

Zunächst wurde die Zustandsraumtransformationemethode vorgeschlagen, die für 2-Stationen-Netze exakte Ergebnisse liefert. Die Idee dabei ist, daß der Teilzustandsraum des Blockiernetzes dem Zustandsraum eines äquivalenten Netzes ohne Blockierung entspricht. Durch die Analyse des äquivalenten Netzes können der Durchsatz, die Auslastung und die Zustandswahrscheinlichkeiten bestimmt werden. Durch die Transformation der Zustandswahrscheinlichkeiten auf Blockiernetze konnten dann die mittleren Auftragszahlen und daraus die anderen Leistungsgrößen ermittelt werden.

Für Mehrstationenfälle konnten wir durch diese Vorgehensweise nur den Durchsatz und die Auslastung bestimmen. In diesem Fall sind die Ergebnisse nicht mehr exakt, sondern sehr gut approximativ (1-10% Abweichung). Der Grund dafür ist der, daß die Zustandsraumtransformation wegen Nichtbijektivität nicht exakt durchführbar ist. Damit können die für das äquivalente Netz ohne Blockierung ermittelten Zustandswahrscheinlichkeiten nicht auf Blockiernetze transformiert werden. Um die mittlere Anzahl von Aufträgen und daraus dann die anderen Leistungsgrößen zu berechnen, wurde die Projektionsmethode vorgeschlagen. Durch diese Methode werden alle nichtausführbaren Zustände, d.h. die Kapazitäten übersteigende Zustände, bzgl. der Kapazitäten normalisiert, m.a.W. werden die Auftragszahlen den Kapazitäten angepaßt. Unter Verwendung der Zustandswahrscheinlichkeiten des Produktformnetzes und der normalisierten Auftragsverteilungen werden dann die mittleren Auftragszahlen ermittelt. Damit werden die Wahrscheinlichkeiten der ausführbaren Zustände im Blockiernetz erhalten. Die dadurch ermittelten mittleren Auftragszahlen weichen von den Simulationswerten um 1-13% ab.

Als zweite Methode für Blockiernetze wurde die Mittelwertanalyse angewandt, die für Produktformnetze große Anerkennung gefunden hat. Dabei wird die MVA so lange durchgeführt, bis die mittlere Anzahl von Aufträgen die Kapazität überschreitet. Im Falle der Überschreitung werden dann die Formeln für die mittlere Verweilzeit modifiziert und die Analyse mit den modifizierten Formeln so lange fortgesetzt, bis die Gesamtzahl K der Aufträge im Netz erhalten ist. Der Algorithmus ist sehr schnell und lie-

fert manchmal erstaunlich gute Ergebnisse. Die großen Abweichungen kommen in den Fällen vor, wo in einem Netz viele Blockiervorgänge auftreten und die Formeln der Mittelwertanalyse immer wieder modifiziert werden müssen. Die Methode kann noch verbessert werden, wenn die Formeln für die mittlere Verweilzeit entsprechend modifiziert werden.

Als letzte Methode für Blockiernetze wurden, basierend auf die Arbeit von /HORD 81/, in der behauptet wird, daß die Blockiernetze Produktformlösungen haben, verschiedene Algorithmen entwickelt. Die Algorithmen beziehen sich auf einen Teil der ausführbaren Zustände des eigentlichen Zustandsraumes. Es wurde festgestellt, daß die Ergebnisse manchmal gut sind. In den Netzen, wo große Abhängigkeit zwischen den Stationen vorliegen, liefert dieses Konzept große Abweichungen.

Ich hatte auch andere Ideen für die Analyse der Blockiernetze, mußte diese jedoch mit der Zeit auf die Seite schieben. Ich denke dabei, z.B. an Phasennetze, d.h. Für ein gegebenes Netz mit endlichen Stationskapazitäten wird ein Netz mit Cox-Phasen entwickelt, wobei aber die Bedienzeiten der 2. Phasen und die Übergangswahrscheinlichkeiten vorher bestimmt und eingesetzt werden. Dabei ergab sich die Schwierigkeit, eine einfache funktionierende Methode für Phasennetze zu finden. Die rekursive Analyse von /SAUE 81/ funktioniert nur für 2 Stationen-Netze und für extrem niedrige Auftragszahlen; wenn mehr Aufträge vorhanden sind, dann erhält man negative Zustandswahrscheinlichkeiten, die die Ergebnisse enorm beeinflussen. Die Methode von /MARI 80/ ist auch aufwendig und in der vorgeschlagenen Form nicht verwendbar. So mußte eine Methode für Phasennetze entwickelt werden. Basierend auf einer bekannten Vorgehensweise wurde eine Methode entwickelt, die sich noch in der Testphase befindet. Mit einem geeigneten Abbruchkriterium erhält man fast exakte Ergebnisse. Es wird noch daran gearbeitet, ein solches Abbruchkriterium zu finden. Wenn die Phasennetze für Blockierungen eingesetzt werden sollen, liegt die andere Schwierigkeit darin, daß die Bedienzeiten der 2. Phasen und die Blockierwahrscheinlichkeiten bestimmt werden müssen. Es wäre möglich, daß diese Größen iterativ ermittelt werden.

Es ist noch zu erwähnen, daß auf der Rechenanlage Cyber 184 des RRZE Erlangen ein Programmpaket QUEBLO in der Programmier-

sprache Pascal unter der Mitarbeit von /PATE 84b/ entwickelt wird. Das Programmpaket enthält die im Kapitel 4 eingeführten Algorithmen für Blockiernetze, wobei später neue Entwicklungen dazu-gefügt werden.

In Kapitel 5 wurden drei Beispiele mit dem vorgeschlagenen Konzept durchgeführt und die Ergebnisse mit denen der Simulation verglichen. Die Simulationsergebnisse wurden aus der Simulation des Gesamtmodells, die im Anhang II ausführlich beschrieben ist, erzielt. Die Vergleiche haben gezeigt, daß die Abweichungen zwischen 1-17% liegen. Das Gesamtmodell wurde mit den im Anhang A1 vorgeschlagenen Algorithmen analysiert. Die Algorithmen wurden in zwei Hauptklassen unterteilt: solche für Produktform- und solche für Nichtproduktformnetze.

Der erste Algorithmus TUPAL bezieht sich auf Produktformnetze und basiert auf der Durchsatzformel. Dieser Algorithmus ist keine überragende neue Idee, ist aber für meine Konzepte in dieser Arbeit, wie leicht festgestellt werden kann, hervorragend anwendbar gewesen, z.B. im Kapitel 4, um die Durchsätze für das äquivalente Netz eines Blockiernetzes zu berechnen. Dieser Algorithmus kann schneller gemacht werden, wenn die in der Formel vorkommende Potenz x_i^{-1} beseitigt oder der Algorithmus in einer Sprache programmiert wird, die die Potenzen als Standardfunktionen enthält (nicht wie in PASCAL).

Im zweiten Algorithmus wurde versucht, den Aufwand der Mittelwertanalyse für lastabhängige Stationen mit $m_i > 1$ durch eine Approximation zu beseitigen. Die Ergebnisse weichen von den exakten um ca. 1-5% ab. Die Vorgehensweise wurde in dieser Arbeit der Übersichtlichkeit halber nur für eine Auftragsklasse durchgeführt. Man kann die Formeln sehr leicht auf Mehrklassenfälle umschreiben. Die Tests wurden auch für Netze mit mehreren Auftragsklassen durchgeführt und ergaben sehr gute Ergebnisse.

In der Reduktionsmethode wurde das Konzept der Parametrischen Analyse erweitert und gezeigt, daß man ein beliebig vorgegebenes Produktformnetz beliebig in zwei Teile zerlegen, diese unabhängig voneinander analysieren und zum Schluß Ergebnisse für das Gesamtmodell erhalten kann. Das Konzept wurde bewiesen und für seine Realisierung der in Abschnitt A1.1.1 eingeführte Algorithmus benutzt. /PATE 84a/ hat gezeigt, daß ein Netz nicht nur

in zwei, sondern in beliebig viele Teile, die mehr als eine Station enthalten dürfen, zerlegt werden kann, wobei diese Teile unabhängig voneinander analysiert werden. Dieses Konzept kann so auf Nichtproduktformnetze erweitert werden, daß die Stationen mit Produktformlösungen zu einem Teil zusammengefaßt und die restlichen im anderen Teil analysiert werden. Dadurch können die Fehlerabweichungen reduziert und die Ergebnisse schneller erzielt werden.

Im Teil der Nichproduktformnetze wurde die Mittelwertanalyse basierend auf einem Vorschlag von /SAUE 81/ aus der LBANC-Methode abgeleitet und modifiziert, um gute approximative Werte zu erhalten. Der Algorithmus EMVA liefert für Erlang-verteilte Bedienzeiten gute Ergebnisse mit Abweichungen von 1-5%. Der Algorithmus ist sehr schnell, wobei man aber berücksichtigen muß, daß die die 100% -Grenze überschreitenden Auslastungswerte immer wieder auf 100% zurückgesetzt werden müssen. Für die Fälle mit hyperexponentiellen Bedienzeitverteilungen liegen die Abweichungen zwischen 8-25%. Der letzte Algorithmus in Abschnitt A1.2, der SIMVAG, der auf einer Idee von /SCHW 79/ für Produktformnetze basiert, bringt eine Verbesserung der Rechenzeit von EMVA.

An dieser Stelle möchte ich noch darauf hinweisen, daß auf der Rechenanlage VAX 11/780 unter der Mitarbeit von /GROS 84/ das Programmpaket ERPE (Erlanger Performance Evaluator) für die Analyse von Warteschlangennetzen in der Programmiersprache PASCAL entwickelt worden ist. Das Programmpaket besteht aus zwei Hauptmenues: exakte und approximative Methoden. Die exakten Methoden wie MVA, LBANC und die von mir entwickelten, in dieser Arbeit im Anhang A1.1 vorgestellten Algorithmen. Die approximativen Verfahren beinhalten die Methode von Marie /MARI 80/, die EPF-Technik /SHUM 76/, die iterative Approximation /CHAN 75b/ und die im Anhang A1.2 eingeführten Algorithmen. Ferner wird eine neue Methode für die Analyse von Warteschlangennetzen mit Phasen hinzugefügt, die zur Zeit noch in der Testphase ist. Es ist auch ein Simulationsteil für das Programmpaket eingeplant, wobei dann die Abweichungen insbesondere für die approximativen Ergebnisse sofort festgestellt werden können.

Das in dieser Arbeit vorgeschlagene Konzept für die hierarchische Analyse funktioniert für die Abarbeitungsstrategie

PS, da alle Prozesse bei sehr kleinen Zeitscheiben simultan aktiv sein können. Es wäre sehr interessant, auch den Fall zu untersuchen, wo die Prozessoren die Abarbeitungsstrategie FCFS haben. In der Simulation des Gesamtmodells wurde auch der Fall der FCFS-Strategie untersucht. Ein Blick auf den Verlauf der Blockierwahrscheinlichkeiten für Prozeß 3, 22, und 27 in einem simulierten System zeigt jedoch, daß die Kurven bei anwachsender Anzahl von Prozessoren keine Gesetzmäßigkeiten erkennen lassen. (vgl. Fig.6.1).

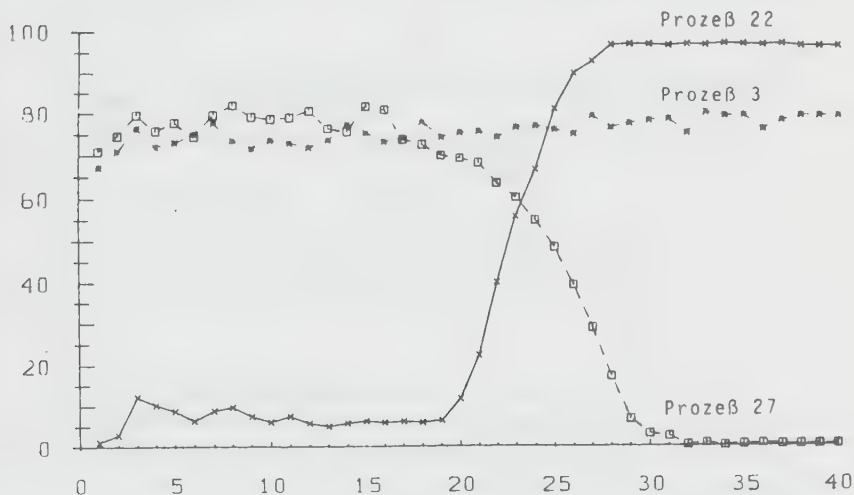


Fig. 6.1 Beispiele für den Verlauf der Blockierwahrscheinlichkeiten in Prozeßkommunikationssystemen

Die in der Fig. 6.1 angegebenen Kurven stellen nur einen kleinen Ausschnitt der Möglichkeiten des Verlaufs der Werte für die Blockierwahrscheinlichkeiten der Prozesse dar. Es wird aber bereits hier deutlich, daß die Werte offensichtlich jeden denkbaren Verlauf nehmen können. Somit könnte ein Konzept zur analytischen Bestimmung der Blockierzeiten und Blockierwahrscheinlichkeiten im Falle der FCFS-Strategie in der Zukunft aufgestellt werden.

A1. ALGORITHMEN FÜR DIE ANALYSE DES GESAMTMODELLS

In Kapitel 5 wurde das Gesamtmodell beschrieben und wurde festgestellt, daß das Modell mit den angenommenen Eigenschaften Produktformlösung hat. Produktformlösung bedeutet, daß die Stationen im Netz unabhängig voneinander analysiert werden können. Im nächsten Abschnitt möchte ich nach einer kurzen Einführung in das Produktform-Konzept einige Algorithmen entwickeln bzw. bereits bestehende Algorithmen verbessern, um den Aufwand zu reduzieren bzw. zusätzliche Stationstypen zu berücksichtigen.

A1.1. Warteschlangennetze mit Produktformlösungen

Ein Netzwerk hat Produktformlösung, wenn jede Bedienstation von einem der folgenden Typen ist:

Typ-1	M/M/m - FCFS
Typ-2	M/G/1 - RR-PS
Typ-3	M/G/IS - .
Typ-4	M/G/1 - LCFS-PR

/JACK 57,63; GORD 67; BCMP 75/ haben gezeigt, daß sich die Wahrscheinlichkeit für einen gegebenen Systemzustand durch das Produkt der Zustandswahrscheinlichkeiten (Randwahrscheinlichkeiten) der einzelnen Stationen ergibt:

$$p(k_1, \dots, k_N) = \frac{1}{G(K)} \prod_{i=1}^N p_i(k_i) \quad (A1.1)$$

wobei k_i die Anzahl der Aufträge in der i-ten Station ist.

$G(K)$ ist die Normalisierungskonstante, die die Summe aller Wahrscheinlichkeiten auf 1 normiert.
(Für offene Netze gilt $G=1$).

$p_i(k_i)$ sind die Randwahrscheinlichkeiten, die anhand der mittleren Bedienzeiten $1/\mu_i$ und den relativen Besuchshäufigkeiten e_i der Aufträge zu der i-ten

Station berechnet werden können.

Die relativen Besuchshäufigkeiten lassen sich aus dem folgenden Gleichungssystem ermitteln:

$$e_i = \sum_{j=1}^N e_j p_{ji} \quad (A1.2)$$

wobei p_{ij} Die Übergangswahrscheinlichkeit von der i-ten zur j-ten Station ist.

Da es im geschlossenen Netz (N-1) unabhängige Gleichungen gibt, nimmt man $e_1=1$ an.

Verschiedene Bedingungen müssen erfüllt sein, damit ein Warteschlangennetz Produktformlösung hat. Diese Bedingungen fassen wir wie folgt zusammen :

a) $M \rightarrow M$ - Eigenschaft /MUNT 73/

Eine Station hat die $M \rightarrow M$ - Eigenschaft genau dann, wenn sie einen Poisson'schen Ankunftsprozess in einen Poisson'schen Abgangsprozess transformiert.

b) "LOCAL-BALANCE" - Eigenschaft /CHAN 72/

Ein Netzwerk ist im lokalen Gleichgewicht genau dann, wenn die Rate, mit der ein Zustand aufgrund des Abgangs eines Auftrags aus einer Station verlassen wird, gleich derjenigen Rate ist, mit der dieser Zustand aufgrund des Übergangs eines Auftrags in diese Station wieder erreicht werden kann.

c) "STATION-BALANCE" - Eigenschaft /CHAN 76/

Eine Warteschlangenstrategie erfüllt die Station-Balance-Eigenschaft, wenn die Rate, mit der Aufträge in einer Position der Warteschlange bedient werden, proportional zur Wahrscheinlichkeit ist, daß ein Auftrag diese Position betritt. Mit anderen Worten wird die Warteschlange einer Station in einzelne Positionen unterteilt und die Raten, mit denen diese Positionen betreten bzw. verlassen werden, wer-

den gleichgesetzt.

/CHAN 76/ zeigen, daß ein Netz, das die Station-Balance-Eigenschaft erfüllt, auch die Local-Balance- und die M M-Eigenschaften erfüllt und Produktformlösung hat. In der gleichen Arbeit wird auch gezeigt, daß Typ-2,3,4-Stationen die Station-Balance-Eigenschaft erfüllen und Typ-1-Stationen nicht. Daher haben Stationen mit FCFS Strategie und nichtexponentieller Bedienzeitverteilung keine Produktformlösungen. Die Beziehungen der Eigenschaften werden in Fig.A1.1. deutlich.

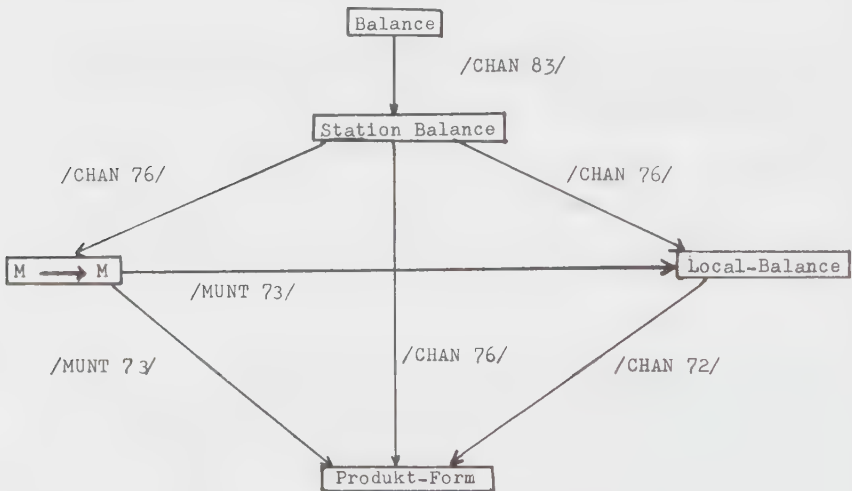


Fig.A1.1.: Beziehungen der Eigenschaften

ERWEITERUNGEN

- Typ-5-Station M/M/1 - SIRO

/SPIR 79/ analysiert die Strategie SIRO (Service in Random Order) und zeigt, daß diese Stationstypen wie Typ 1 Produktformlösungen haben.

- Typ-6-Station M/M/1 - LBPS

/NOET 79/ führt die Strategie LBPS (Last Batch Processor Sharing) ein und zeigt, daß die Lösungen für solche Stationstypen Produktform haben. Bei dieser Strategie wird der Prozessor zwischen den letzten Batch-Aufträgen verteilt. Falls der letzte Batch einen einzigen Auftrag enthält, hat man ICFS-PR; falls er alle Aufträge enthält, hat man PS.

- Typ-7-Station M/M/1 - WEIRDP (und BALANCE-Konzept)

/CHAN 83/ behandelt ein neues Konzept 'BALANCE', mit dem er beweist, daß Preemptive Prioritätsstrategien nicht das Balance-Konzept erfüllen und daher keine Produktformlösung haben können. Das Balance-Konzept bezieht sich auf die Warteschlangenstrategie und existiert dann, wenn eine Strategie eine charakteristische Funktion besitzt, die alle nicht-negativen Integervektoren k (Zustände) als Definitionsbereich hat. Für die Bestimmung der charakteristischen Funktion wird ein iterativer Algorithmus gegeben und mit Hilfe eines Theorems gezeigt, daß die Balance-Eigenschaft in der Station-Balance enthalten ist.

Auf dem Balance-Konzept basierend behandeln /CHAN 83/ die Strategie WEIRDP. Bei dieser Strategie wird dem ersten Auftrag in der Warteschlange ein Teil p des Prozessors und $(1-p)$ den restlichen Aufträgen zugeteilt. Sie geben eine Produktformlösung für solche Stationstypen und zeigen, daß das Konzept von /NOET 79/ richtig ist.

A1.1.1. TUPAL (Throughput-Algorithm)

Bei der Analyse von Warteschlangennetzen werden alle Leistungsgrößen eines Systems mit Hilfe der Zustandswahrscheinlichkeiten (Gl.A1.1) berechnet. Um die Zustandswahrscheinlichkeiten zu berechnen, muß, wie aus Gl.(A1.1) ersichtlich, die Normalisierungskonstante im Falle geschlossener Netze ermittelt werden.

Ein umständlicher Weg für die Berechnung von $G(K)$ ist die Aufzählung aller Zustände und die Bestimmung deren relativer Wahrscheinlichkeiten. Absolute Wahrscheinlichkeiten können dann aus den relativen Wahrscheinlichkeiten durch Normierung von deren Summe zu 1 bestimmt werden. Dies ist natürlich nur für kleine Netze durchführbar, weil für größere Netze die Anzahl der Zustandswahrscheinlichkeiten auch dementsprechend zunimmt.

/BUZE 71/ hat einen effizienten Algorithmus für $G(K)$ entwickelt, ohne die Zustände des Netzes in die Berechnung einzubeziehen. Mit diesem Algorithmus ist es möglich, die Leistungsgrößen mit einfachen Formeln zu ermitteln. Inzwischen gibt es mehrere, zum Teil sehr unterschiedliche Algorithmen zur Bestimmung der Normalisierungskonstanten. Einer von den Algorithmen, die große Anerkennung gefunden haben, ist der Faltungsalgorithmus (= Convolution Algorithm) von /BUZE 71,73; CHAN 75a/, der von /WONG 75/ auf Mehrklassennetze erweitert und von /LIEN 83/ zur optimalen Speicherausnutzung zum Baum-Faltungsalgorithmus modifiziert wurde. Der andere populäre einfache Algorithmus ist der LBANC von /SAUE 81/, der von der MVA abgeleitet wurde.

Da die Normalisierungskonstante Over-/Underflow-Probleme mit sich bringt, hat man versucht, die Leistungsgrößen ohne Zuhilfenahme der Normalisierungskonstanten zu berechnen. Dabei entstand die MVA /REIS 79,80,81/, die in den letzten Jahren große Anerkennung gefunden hat. Mit der MVA lassen sich iterativ unter Verwendung von drei fundamentalen Gleichungen für die mittlere Verweilzeit, den Durchsatz und die mittlere Anzahl von Aufträgen die Leistungsgrößen bestimmen. Sie ist leicht implementierbar und ist sehr schnell. Für Netze mit lastabhängigen Stationen ($m > 1$) und mehreren Auftragsklassen hat sie jedoch große Nachteile, wie z.B. Speicherplatz-, Stabilitäts- und Rundungsfehlerprobleme.

Im folgenden Algorithmus zeige ich, daß die Iteration für die

Bestimmung der Leistungsgrößen nur über eine einzige Formel (Durchsatz-Formel) durchgeführt und daraus dann alle anderen interessierenden Leistungsgrößen bestimmt werden können.

Theorem

In einem geschlossenen Warteschlangennetz mit Typ-1,2,3,4,5,6,7-Stationen ergibt sich der Durchsatz des Systems aus der folgenden Formel:

$$\lambda(K) = \begin{cases} K \cdot \left[\begin{matrix} K & 1-1 \\ \sum_{l=1} & \Pi_{j=1} \end{matrix} \lambda(K-j) \begin{matrix} N \\ \sum_{i=1} x_i \end{matrix} \right]^{-1} & \text{Typ 1,2,4,5,6,7} \\ K \cdot \left[\begin{matrix} K-1 & & N & K \\ \Pi_{j=1} & \lambda(j) & \sum_{i=1} & \sum_{l=1} \end{matrix} 1 \cdot p_i(1/K) \right]^{-1} & \text{für Typ 1 (} m_i \geq 1 \text{)} \\ K \cdot \left[\begin{matrix} N \\ \sum_{i=1} x_i \end{matrix} \right]^{-1} & \text{Typ 3} \end{cases} \quad (A1.3)$$

wobei $x_i = \frac{e_i}{\mu_i}$ die relative Auslastung der i-ten Station ist.
 e_i ermittelt man aus Gl.(A1.2).

$p_i(1/K)$ ist die Wahrscheinlichkeit, daß in der i-ten Station 1 Aufträge bedient werden, unter der Bedingung, daß sich im Gesamtsystem K Aufträge befinden. Sie ergibt sich aus :

$$p_i(1/K) = \frac{x_i \cdot p_i(1-1/K-1)}{\gamma_i(1)} \quad (A1.5)$$

mit

$$p_i(0/K) = \left[\prod_{j=1}^K \lambda(j) \right]^{-1} - \sum_{l=1}^K p_i(l/K) \quad (A1.6)$$

$$p_i(0/0) = 1 \quad p_i(1/0) = 0 \quad \forall i = 1, \dots, N$$

Die Funktion $\gamma_i(l)$ berechnet man aus der Formel:

$$\gamma_i(l) = \begin{cases} 1 & \text{für } l \leq m_i \\ m_i & \text{für } l \geq m_i \end{cases} \quad (A1.7)$$

Als Initialwert nimmt man

$$\lambda(1) = \frac{1}{\sum_{i=1}^N x_i}$$

Der Algorithmus bricht dann ab, wenn die Gesamtzahl der Aufträge K erreicht ist.

Beweis

1) für $m_i=1$:

Man geht aus von der Formel für den Durchsatz,

$$\lambda(K) = \frac{G(K-1)}{G(K)} \quad (A1.8)$$

für die Normalisierungskonstante

$$G(K) = \frac{1}{\prod_{j=1}^K \lambda(j)} \quad (A1.9)$$

und

$$G(K) = \frac{1}{K} \sum_{i=1}^N \sum_{l=1}^K x_i^l G(K-l) \quad (A1.10)$$

Die Gleichung (A1.10) kann aus der Formel für die mittlere Anzahl von Aufträgen in der i-ten Station abgeleitet werden
/BOAK 82/

$$\bar{k}_i(K) = \sum_{l=1}^K x_i^l \frac{G(K-l)}{G(K)}$$

Beide Seiten der Gleichung werden von $i=1, \dots, N$ summiert:

$$\sum_{i=1}^N \bar{k}_i(K) = \sum_{i=1}^N \sum_{l=1}^K x_i^l \frac{G(K-l)}{G(K)}$$

Da die Anzahl der Aufträge in einem geschlossenen Netz konstant ist, erhält man

$$K = \sum_{i=1}^N \sum_{l=1}^K x_i^l \frac{G(K-l)}{G(K)}$$

Nach Umformung nach $G(K)$ erhalten wir Gl.(A1.10).

Nun wieder zurück zum eigentlichen Beweis.

Durch Einsetzen von Gl.(A1.9) in Gl.(A1.10) erhält man

$$G(K) = \frac{1}{K} \sum_{i=1}^N \sum_{l=1}^K x_i^l \frac{1}{\prod_{j=0}^{K-l} \lambda(j)} \quad (A1.11)$$

Setzt man in Gl.(A1.8) für den Zähler Gl.(A1.9) und für den Nenner Gl.(A1.11) ein, so bekommt man

$$\lambda(K) = \frac{1}{\prod_{j=0}^{K-1} \lambda(j)} \left[\frac{1}{K} \sum_{i=1}^N \sum_{l=1}^K x_i^l \frac{1}{\prod_{j=0}^{K-1} \lambda(j)} \right]^{-1} \quad (A1.12)$$

Das Produkt von $j=0, \dots, K-1$ über $\lambda(K)$ kann unter die Summenzeichen gezogen werden :

$$\lambda(K) = \frac{K}{\sum_{i=1}^N \sum_{l=1}^K x_i^l \left[\prod_{j=0}^{K-1} \lambda(j) / \prod_{j=0}^{K-1} \lambda(j) \right]} \quad (A1.13)$$

Die ersten $(K-1)$ Faktoren der Produkte fallen durch Kürzungen weg :

$$\lambda(K) = \frac{K}{\sum_{i=1}^N \sum_{l=1}^K x_i^l \prod_{j=K-l+1}^{K-1} \lambda(j)} \quad (A1.14)$$

Damit erhält man Gl.(A1.3) für $m_i=1$.

ii) Für $m_i \geq 1$:

Nach /SAUE 81/ gilt:

$$G(K) = \frac{1}{K} \sum_{i=1}^N \sum_{l=1}^K l p_i(l/K)$$

Durch Einsetzen der Gl.(A1.14) in Gl.(A1.9) und Umformung erhält man Gl.(A1.3) für $m_i \geq 1$.

iii) Für $m_i = \infty$:

In diesem Fall gilt :

$$\bar{t}_i = \frac{1}{\mu_i}$$

d.h. mittlere Verweilzeit = mittlere Bedienzeit

Nach Littles Gesetz gilt :

$$\bar{t}_i = \frac{e_i \lambda(K)}{\mu_i}$$

Durch Summierung beider Seiten über $i=1, \dots, N$ und Berücksichtigung, daß die Anzahl der Aufträge in einem geschlossenen Netz konstant ist, erhalten wir dann Gl.(A1.3) für $m_i = \infty$

q.e.d.

Aus Gl.(A1.3) für den Durchsatz können alle anderen Leistungsgrößen berechnet werden :

Die mittlere Anzahl von Aufträgen :

$$\bar{k}_i(K) = \sum_{l=0}^{K-1} \prod_{j=0}^l \lambda(K-j) x_i^{l+1} \quad (A1.15)$$

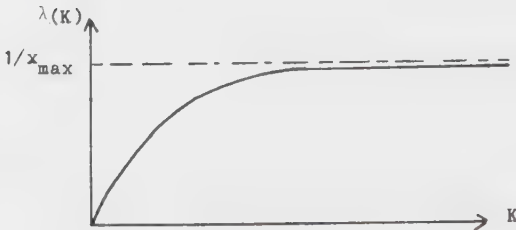
Die Zustandswahrscheinlichkeiten :

$$p(k_1, \dots, k_N) = \prod_{i=1}^N x_i^{k_i} \prod_{l=0}^{k_i} \lambda(k_i - l) \quad (A1.16)$$

Dies folgt direkt aus der Berücksichtigung der Gl.(A1.9).

APPROXIMATION VON TUPAL

Eine wesentliche Rechenzeitverkürzung erreicht man, wenn man auf exakte Lösungen verzichtet und sie nur approximiert. Dabei nutzt man die Tatsache, daß der Durchsatz in Abhängigkeit von der Anzahl von Aufträgen im System gegen einen Maximalwert konvergiert (siehe Fig.A1.2).



Dieser Maximalwert $\lambda(K) = 1/x_{\max}$ mit $x_{\max} = \max(x_i)$, für $i=1, \dots, N$ wird im allgemeinen schon nach relativ wenigen Iterationen hinreichend genau erreicht, sodaß die Berechnung dann abgebrochen werden kann. Man beginnt die Iteration mit $k=1$ und bestimmt immer wieder den Durchsatz für die nächsthöhere Anzahl von Aufträgen im System. Nach jeder Iteration wird geprüft, ob

$$|\lambda(k) - \lambda(k-1)| < \epsilon.$$

Falls die Terminationsbedingung erfüllt wird, bricht der Algorithmus ab. Die Durchsätze für die höhere Anzahl von Aufträgen werden dann dem letzten Durchsatzwert gleichgesetzt. Die Erfahrungen haben gezeigt, daß die dadurch erhaltenen Ergebnisse sehr gut sind. Sie unterscheiden sich von den exakten Werten meist erst in der dritten Stelle nach dem Komma.

Bewertung

Der Algorithmus TUPAL und seine Approximation wurden auf der Rechenanlage VAX 11/780 in der Programmiersprache PASCAL implementiert. Der Algorithmus wurde mit den bekannten herkömmlichen Methoden wie z.B. Mittelwertanalyse und LBANC-Methode verglichen.

Es wurden ca. 30 numerische Beispiele durchgeführt und die Rechenzeiten gemessen. Bei den Testläufen wurden die CPU-Zeiten mit Hilfe einer internen Funktion in der Programmiersprache C gemessen. Diese Funktion liefert als Wert die aktuelle Uhrzeit in Form einer real-Größe zurück. Mit Hilfe von 2 Variablen im Programm kann die Uhrzeit an 2 Stellen bestimmt und die CPU-Zeit als Differenz dieser beiden Variablen berechnet werden.

Für die von den Algorithmen verbrauchte CPU-Zeit ergaben sich folgende Werte in msec :

Alg K	TUPAL	LBANC	MVA
10	330	270	130
25	410	700	210
50	550	870	220
75	660	990	240
100	710	1010	300
125	920		330

Wie aus der Tabelle ersichtlich, ist die Mittelwertanalyse der schnellste Algorithmus. Der Grund dafür, daß TUPAL langsamer ist als die MVA, liegt vor allem darin, daß in TUPAL die Potenz (x_1^{-1}) vorkommt. Das Vorkommen der Potenz erhöht die Rechenzeit beträchtlich, wenn in Programmiersprachen gearbeitet wird, die sie nicht als Standardroutine bieten, sondern als Unterprogramm verlangen wie bei PASCAL. Im Vergleich zur LBANC ist der TUPAL schneller. Die LBANC-Methode liefert der Wert für die Normalisierungskonstante ab $K=125$ nicht mehr, weil der Wert bereits für $K=100$ bei etwa $3.6 \cdot 10^{31}$ lag.

Als Grundlage für den Speicherplatzbedarf wird die Anzahl der

Speicherplätze angenommen, die als Zwischenspeicher maximal reserviert werden müßten.

- TUPAL $O(K_R)$
- LBANC $O(N \cdot \prod_{r=1}^R (K_r+1))$
- MVA $O(N \cdot \prod_{r=1}^R (K_r+1))$

Hinsichtlich des Speicherplatzbedarfes ist der neue TUPAL-Algorithmus den herkömmlichen Methoden überlegen.

Bei einem Vergleich des Speicherplatzbedarfes muß allerdings immer berücksichtigt werden, daß durch die unterschiedlichen Möglichkeiten der Programmierung mehr oder weniger Speicherplatz eingespart werden kann.

A1.1.2. APPROXIMATION DER MITTELWERTANALYSE FÜR LASTABHÄNGIGE STATIONEN

Die Mittelwertanalyse, kurz MVA (Mean Value Analysis), ist eine exakte Methode zur Analyse von Produktformwarteschlangennetzen.

Sie basiert auf zwei Theoremen:

1.Theorem (Verteilung beim Ankunftszeitpunkt)

In einem geschlossenen Warteschlangennetz ist die Wahrscheinlichkeit, daß ein Auftrag, der in die Station i eintritt, den Netzwerkzustand $\underline{k} = (k_1, \dots, k_i, \dots, k_N)$ vorfindet, gleich der Zustandswahrscheinlichkeit $p(k_1, \dots, k_i-1, \dots, k_N)$ des Systems im Gleichgewicht mit einem Auftrag weniger.

/SEVC 81, REIS 80/ haben dieses Theorem bewiesen und haben gezeigt, daß es für alle Netze mit Produktformlösung gilt.

2.Theorem (Das Theorem von Little)

Mittlere Anzahl von Aufträgen = Durchsatz $\cdot$ Mittlere Verweilzeit

$$\bar{k} = \lambda \cdot \bar{t}$$

Der Beweis kann in /SAUE 81/ nachgelesen werden.

Die Mittelwertanalyse für $m_i \geq 1$ wird wie folgt durchgeführt:

a) Initialisiere

$$\begin{array}{ll} \bar{k}_i(0) = 0 ; & p_i(0/0) = 1 ; \\ p_i(1/0) = 0 & \text{für } l=1, \dots, m_i-1; \quad i=1, \dots, N \end{array}$$

b) Iteriere $k = 1, \dots, K$

1) Die mittlere Verweilzeit folgt aus dem 1.Theorem:

$$\bar{t}_i(k) = \frac{1}{\mu_i} \left[1 + \bar{k}_i(k-1) + \sum_{l=1}^{m_i-2} (m_i-1-l) p_i(l/k-1) \right] \quad (A1.17)$$

wobei die bedingten Wahrscheinlichkeiten $p_i(l/k)$
(vgl. Gl.(A1.5)) aus den folgenden Gleichungen
ermittelt werden können.

$$p_i(l/k) = x_i \frac{\lambda(k)}{1} p_i(l-1/k-1) \text{ für } l=1, \dots, m_i-1 \quad (A1.18)$$

$$p_i(0/k) = 1 - \frac{1}{m_i} \left[x_i \lambda(k) + \sum_{l=1}^{m_i-1} (m_i-l) p_i(l/k) \right] \quad (A1.19)$$

ii) Der Durchsatz folgt aus dem 2.Theorem:

$$\lambda(k) = \frac{k}{\sum_{i=1}^N e_i \cdot \bar{t}_i(k)} \quad (A1.20)$$

Die Durchsätze einzelner Stationen ergeben sich aus

$$\lambda_i = \lambda(k) \cdot e_i \quad \text{für } i=1, \dots, N \quad (A1.21)$$

wobei e_i mit Gl.(A1.2) ermittelt wird.

iii) Die mittlere Anzahl von Aufträgen folgt auch aus dem
2.Theorem:

$$\bar{k}_i(k) = \lambda(k) \cdot \bar{t}_i(k) \cdot e_i \quad \text{für } i=1, \dots, N \quad (A1.22)$$

Diese Gleichungen ((A1.17) - (A1.22)) erlauben eine iterative Bestimmung der Leistungsgrößen, wobei der Algorithmus genau dann abbricht wenn K , die Gesamtzahl der Aufträge im Netz, erreicht ist.

Wenn darüber hinaus Zustandswahrscheinlichkeiten in einem Netz bestimmt werden sollen, können wir die MVA um einen einzigen einfachen Zwischenschritt erweitern, der es ermöglicht, die

Normalisierungskonstante und dadurch ohne großen Aufwand die Gleichgewichtszustandswahrscheinlichkeiten zu ermitteln.

Die Normalisierungskonstante läßt sich in der MVA durch folgende Formel iterativ bestimmen /AKYL 83/:

$$G(k) = \frac{G(k-1)}{\lambda(k)} \quad \text{mit } G(0) = 1 \quad (A1.23)$$

Die Mittelwertanalyse geht davon aus, daß die Leistungsgrößen für die Anzahl von Aufträgen $(k-1)$ bekannt sind. Die Iteration wird ausgehend von der Anzahl von Aufträgen 0 bis hin zur Gesamtzahl K durchgeführt. Dieses Prinzip macht den MVA-Algorithmus so extrem aufwendig; insbesondere bei der Analyse von Warteschlangennetzen mit mehreren Auftragsklassen entstehen erhebliche Speicherplatzprobleme. Außerdem kommen im Falle lastabhängiger Stationen ($m_i > 1$) bei den Iterationen (z.B. ab 60. Iteration) negative Werte für bedingte Randwahrscheinlichkeiten $p_i(1/k)$ vor, die dann auf 0 normiert werden müssen. Damit sind die erhaltenen Leistungsgrößen nicht mehr exakt. Wie schon oben erwähnt wurde, besteht das Problem bei der MVA darin, daß sie eine komplette Lösung von 0 bis K Aufträgen im System erfordert.

Ein alternativer Ansatz kann darin bestehen, daß man aufgrund der Information über die Anzahl der Aufträge K Leistungsgrößen der Anzahl von Aufträgen $(K-1)$ abschätzt und dann die MVA anwendet, um die Leistungsgrößen für K zu berechnen.

Um die Leistungsgrößen bei $(K-1)$ zu schätzen, ging /SCHW 79/ von der folgenden Formel aus :

$$\bar{k}_i = \frac{K}{N}$$

als Initialwert und

$$\bar{k}_i(K-1) = \frac{K-1}{K} \bar{k}_i \quad (A1.24)$$

Da wir aber in der Formel für die mittlere Verweilzeit (Gl.A1.17) nicht nur $\bar{k}_i$ -Größen, sondern auch die bedingten Randwahrscheinlichkeiten $p_i(1/k)$ haben, müssen wir zusätzlich noch Schätzungen dieser bedingten Wahrscheinlichkeiten bei der Anzahl von Aufträgen $(K-1)$ machen.

/NEUS 81/ haben einen Ansatz (SCAT-Algorithmus) vorgeschlagen, der die Wahrscheinlichkeitsmasse so nahe wie möglich an die mitt-

lere Anzahl von Aufträgen legt. Das bedeutet also: wenn z.B. die mittlere Anzahl von Aufträgen k_1 ($K-1$) = 2.6 ist, dann wird geschätzt, daß die Wahrscheinlichkeit, daß sich 2 Aufträge in der Station i befinden, $p_i(2/K-1) = 0.4$ und die Wahrscheinlichkeit von 3 Aufträgen $p_i(3/K-1) = 0.6$ ist.

Formal wird dieser Ansatz durch folgende Formeln dargestellt (die Floor-Notation stammt von /NEUS 81) :

$$\text{Floor}_i = \lfloor \bar{k}_i(K-1) \rfloor$$

$$p_i(\text{Floor}_i | K-1) = \text{Floor}_i + 1 - \bar{k}_i(K-1) \quad (\text{A1.25})$$

$$p_i(\text{Floor}_i + 1 | K-1) = 1 - p_i(\text{Floor}_i | K-1)$$

$$p_i(l | K-1) = 0 \quad \text{für } l < \text{Floor}_i \text{ \& } l > \text{Floor}_i + 1$$

Dieser Ansatz entspricht jedoch im allgemeinen nicht den wirklichen Verhältnissen in Produktform-Warteschlangennetzwerken. Die Untersuchung einer großen Anzahl von Netzen, die aus Stationen mit mehreren Bedieneinheiten bestanden, haben gezeigt, daß die Fehlerabweichungen manchmal weit über 30% liegen.

Dies kommt vor, wenn die Fehlertoleranz wie folgt definiert wird:

$$F = \frac{\bar{k}_i^e(k) - \bar{k}_i^a(k)}{\bar{k}_i^e(k)}$$

wobei e exakt und a approximativ bezeichnen.

/NEUS 81/ definiert jedoch die folgende Fehlertoleranzformel:

$$\text{Tol} = \frac{\bar{k}_i^e(\underline{k}) - \bar{k}_i^a(\underline{k})}{k}$$

was natürlich bewirkt, daß die Fehlerabweichungen, die in Wirklichkeit 30% waren, werden plötzlich auf 2% heruntergesenkt. Deshalb wird hier eine neue, verbesserte Schätzung der Randwahrscheinlichkeiten entwickelt. Diese Schätzung wird im nächsten Abschnitt gesondert behandelt.

Eine verbesserte Schätzung der bedingten Randwahrscheinlichkeiten

Der Ansatz, die Wahrscheinlichkeitsmasse so nah wie möglich an die mittlere Anzahl von Aufträgen in der i-ten Station $\bar{k}_i(K-1)$ zu legen, berücksichtigt in keiner Weise, daß es sich bei den $\bar{k}_i$ -Werten um Mittelwerte handelt. Ist z.B. $\bar{k}_i(K-1) = 2.6$, so sind die Wahrscheinlichkeiten, daß sich 0, 1, 4 und 5 Aufträge in der i-ten Station befinden, auf keinen Fall zu vernachlässigen.

Da die Randwahrscheinlichkeiten $p_i(l/K-1)$ bei der Berechnung der Verweilzeiten nur für $l=0, \dots, m_i-2$ betrachtet werden (siehe Gl.(A1.17)), treten besonders in denjenigen Fällen gravierende Fehler auf, in denen sich die $\bar{k}_i$ -Werte in der Größenordnung der Anzahl identischer Bedieneinheiten in der Station m_i bewegen, weil in diesen Fällen die relevanten Wahrscheinlichkeiten $p_i(l/K-1)$, $l=0, \dots, m_i-2$ gleich 0 gesetzt werden.

Der Ansatz für die Schätzung muß also genau umgekehrt werden, d.h. die Wahrscheinlichkeiten müssen so breit wie möglich gestreut werden. Die hier vorgeschlagene Schätzung geht wie die von /NEUS 81/ von den $\bar{k}_i(K-1)$ -Werten aus. Die Wahrscheinlichkeitsmasse wird symmetrisch so auf die oberhalb und unterhalb des geschätzten $\bar{k}_i(K-1)$ -Wertes liegenden Auftragszahlen verteilt, daß das Auftreten von $(\text{Floor}_i - m)$ -Aufträgen und das von $(\text{Floor}_i + 1 + m)$ -Aufträgen, bei der $(K-1)$ im Mittel $\bar{k}_i(K-1)$ ergibt.

Formal ausgedrückt :

$$\bar{k}_i(K-1) = \frac{1}{W(S, m)} \left[p_i(\text{Floor}_i - m/K-1) \cdot (\text{Floor}_i - m) + p_i(\text{Floor}_i + 1 + m/K-1) \cdot (\text{Floor}_i + 1 + m) \right] \quad (\text{A1.26})$$

wobei $m = 0, \dots, S$

$$S = \min \{ \text{Floor}_i; K-1-1-\text{Floor}_i \} \quad (\text{A1.27})$$

$$W(S,m) = p_i(\text{Floor}_i - m/K-1) + p_i(\text{Floor}_i + 1 + m/K-1) \quad (\text{A1.28})$$

$$\sum_{m=0}^S W(S,m) = 1 \quad (\text{A1.29})$$

S bezeichnet also die maximale Streubreite, die wegen der symmetrischen Verteilung der Wahrscheinlichkeitsmasse maximal wird, wenn

$$\text{Floor}_i = \lfloor |K-1|/2 \rfloor,$$

also in der Mitte zwischen 0 und der Gesamtzahl der Aufträge bei der Anzahl der Aufträge im System $(K-1)$ liegt. Ist die Streubreite S, so wird die Wahrscheinlichkeitsmasse auf $(S+1)$ Zahlenpaare verteilt.

$W(S,m)$ bezeichnet die dem $(m+1)$ -ten Zahlenpaar zugeordnete Wahrscheinlichkeitsmasse bei einer Verteilung der Gesamtmasse auf $(S+1)$ Zahlenpaare.

Aus den Gleichungen (A1.26) und (A1.29) folgt:

$$p_i(\text{Floor}_i - m/K-1) = \frac{W(S,m)}{2m+1} \left[(\text{Floor}_i + 1 + m - \bar{K}_i)(K-1) \right]$$

und

$$p_i(\text{Floor}_i + 1 + m/K-1) = W(S,m) - p_i(\text{Floor}_i - m/K-1) \quad (\text{A1.30})$$

$$p_i(1/K-1) = 0 \quad \text{für } 1 < \text{Floor}_i - S \wedge 1 > \text{Floor}_i + 1 + S$$

Im Falle $S=0, l=0$ entspricht diese Gleichung der Verteilungsformel (A1.25) von /NEUS 81/.

Auf das obige Beispiel mit $\bar{K}_i(K-1) = 2.6$ bezogen, bedeutet diese Verteilung folgendes:

Unter der Annahme, daß die dem ersten Zahlenpaar (2,3) zugeteilte Wahrscheinlichkeitsmasse bei einer Verteilung der Gesamtmasse auf nur 2 Zahlenpaare (2,3) und (1,4) $W(1,0) = 0.6$ beträgt, erhält man für dieses Paar die Randwahrscheinlichkeiten:

$$p_i(2|K-1) = \frac{0.6}{1} (2+1-2.6) = 0.24$$

$$p_i(2|K-1) = 0.6 - 0.24 = 0.36$$

und für das Paar (1,4) entsprechend

$$p_i(1|K-1) = \frac{0.4}{3} (2+1+1-2.6) = 0.187$$

$$p_i(4|K-1) = 0.4 - 0.187 = 0.213$$

Diese Verteilung (Gl. A1.30) hat jedoch noch den Nachteil, daß die Tatsache, daß $p_i(0|K-1)$ nie gleich 0 sein kann (Gl.(A1.19)), in den Fällen, in denen

$$S = \min(\text{Floor}_i; |K-1|-1-\text{Floor}_i) = |K-1|-1-\text{Floor}_i$$

ist, nicht berücksichtigt wird. D.h. liegt Floor_i oberhalb des Wertes $(|K-1|/2)$, so können bei kleinen Anzahlen von Aufträgen Extremfälle auftreten, die zu großen Fehlerabweichungen führen, weil die Streubreite zu klein ist.

Modifikation der Verteilung

Deshalb wird diese Verteilung noch modifiziert, indem die Streubreite S immer gleich Floor_i gesetzt wird. Tritt nun der gerade geschilderte Fall $\text{Floor}_i > |K-1|/2$ ein, so werden die Randwahrscheinlichkeiten $p_i(0|K-1), \dots, p_i(2*\text{Floor}_i-(K-1)|K-1)$ so zu der Wahrscheinlichkeit $p_i(|K-1|/K-1)$ ins Verhältnis gesetzt, daß das Auftreten dieser Auftragszahlen in der Station zusammen wieder den Mittelwert $\bar{k}_i(K-1)$ ergibt.

Formal

$$p_i(l|K-1) = \frac{W(S, \text{Floor}_i-1)}{|K-1|-1} (|K-1|-\bar{k}_i(K-1)) \quad (\text{A1.31})$$

für $l=0, \dots, L$ mit $L = 2*\text{Floor}_i - K-1$.

Für $l=L, \dots, K-1$ werden die Randwahrscheinlichkeiten nach Gl.(A1.30) berechnet.

Das Problem besteht nun darin, die Wahrscheinlichkeitsmasse auf die einzelnen Zahlenpaare zu verteilen. Mein Ansatz geht dahin, daß die prozentuale Abnahme der Wahrscheinlichkeitsmasse bei zunehmender Streuung für alle Zahlenpaare gleich groß ist.

Wenn also in unserem Beispiel dem Zahlenpaar (2,3) die Wahr-

scheinlichkeitsmasse $W(1,0) = 0.6$ und dem Zahlenpaar $(1,4)$ $W(1,4) = 0.4$ zugeteilt sind, so verringern sich diese Werte bei einer Streuung auf 3 Zahlenpaare ($S=2$) gleichmäßig um z.B. 10%; das bedeutet:

$$W(2,0) = 0.6 - 0.6 \frac{10}{100} = 0.54$$

$$W(2,1) = 0.4 - 0.4 \frac{10}{100} = 0.36$$

$$W(2,2) = 1 - [W(2,0) + W(2,1)] = 0.1$$

Formal

$$W(0,0) = 1$$

$$W(n,m) = W(n-1,m) - W(n-1,n) - \frac{PR(n)}{100} \quad (A1.32)$$

für $n=1, \dots, S$ und $m=0, \dots, n-1$

$$W(n,n) = 1 - \sum_{m=0}^{n-1} W(n,m)$$

Dabei bezeichnet $PR(n)$ die Abnahme der Wahrscheinlichkeitsmasse in Prozent. $PR(n)$ sollte eine Funktion sein, deren Funktionswerte mit zunehmender Streuung der Randwahrscheinlichkeiten abnehmen, damit die Wahrscheinlichkeitsmassen zum Mittelwert $\bar{k}_1(K-1)$ hinzunehmen. Deshalb sollte der Startwert $PR(1) < 50$ sein, damit der größte Anteil der gesamten Wahrscheinlichkeitsmasse dem ersten Zahlenpaar zugeordnet wird.

$$PR(1) = 45$$

$$PR(n) = PR(n-1) * 0.7 \quad \text{für } n > 1 \quad (A1.33)$$

Der Initialwert $PR(1) = 45$ und der Verminderungsfaktor 0.7 sind empirische Größen. Es gibt auch Stationen, deren Randwahrscheinlichkeiten besser mit $PR(1) = 40$ approximiert werden können, aber besonders in den kritischen Bereichen ($\bar{k}_1(K-1) \approx m_1$) hat sich der obige Wert durchgesetzt.

Ebenso empfiehlt sich die Einführung einer Obergrenze für die Streubreite, damit der Berechnungsaufwand der Algorithmen nicht mit steigenden Auftragszahlen anwächst. Gute Erfahrungen habe ich mit einer maximalen Streubreite $S_{\max} = 20$ gemacht. Diese Streubreite reicht vollkommen aus, da bei einer Verteilung der Wahrscheinlichkeitsmasse auf 21 Zahlenpaare dem 21. Paar nur noch $W(20,20) = 0.0007$ zugeteilt wird.

Die maximale Streubreite berechnet sich dann wie folgt :

$$S_{\max} = \min (20 ; |K-1|) \quad (A1.34)$$

Betrachten wir noch einmal das Beispiel mit $\bar{k}_i(K-1) = 2.6$ und nehmen wir an, daß sich bei $(K-1)$ Aufträgen mehr als 4 Aufträge im Netzwerk befinden. Dann ergibt sich die Breite der Streuung S mit Gl.(A1.27) zu $S=2$.

Zuerst wird also ausgehend von $PR(1) = 45$ iterativ die Abnahme der Wahrscheinlichkeitsmasse in Prozent bestimmt nach Gl.(A1.33) :

$$PR(1) = 45$$

$$PR(2) = 31.5$$

Mit diesen Werten wird dann die Wahrscheinlichkeitsmassenverteilung berechnet nach Gl.(A1.32).

$$W(0,0) = 1$$

$$W(1,0) = 0.55 \quad W(1,1) = 0.45$$

$$W(2,0) = 0.3767 \quad W(2,1) = 0.3083 \quad W(2,2) = 0.315$$

Setzt man diese Werte in Gl.(A1.30) ein, so erhält man endlich auch die Randwahrscheinlichkeiten :

$$p_i(0/K-1) = 0.1512$$

$$p_i(1/K-1) = 0.1439$$

$$p_i(2/K-1) = 0.1507$$

$$p_i(3/K-1) = 0.2260$$

$$p_i(4/K-1) = 0.1644$$

$$p_i(5/K-1) = 0.1638$$

Wir können die Gesamtvorgehensweise wie folgt zusammenfassen:

1. Schritt

- Berechne die maximale Streubreite mit Gl.(A1.34)
- Berechne die prozentuale Abnahme der Wahrscheinlichkeitsmasse $PR(n)$ mit Gl.(A1.33)
- Berechne die Verteilung der Wahrscheinlichkeitsmasse $W(S,m)$ mit Gl.(A1.32)

2. Schritt

- Berechne den Initialwert $K_i = \frac{K}{N}$
- Berechne die Schätzungen der Randwahrscheinlichkeiten $p_i(l/K-1)$ für $l=0, \dots, m_i-2$ aus Gl.(A1.24), (A1.30) und (A1.31) für jede lastabhängige Station i .

3. Schritt

- Berechne neue k_i -Werte ausgehend von Gl.(A1.17), (A1.20) und (A1.22).

4. Schritt Terminationstest

- if $|\bar{k}_i(K) - \bar{k}_i(K-1)| < \epsilon$
 then STOP
 else goto 1.Schritt

Bewertung

Diese Vorgehensweise wurde in der Programmiersprache PASCAL programmiert und auf einer CDC CYBER 173 realisiert.

Die Genauigkeit dieser Vorgehensweise wurde durch Experimente bestimmt. Die Experimente sahen so aus, daß eine Reihe von Netzwerken ausgewählt wurde, in denen die Anzahl der Aufträge von 4 bis 200 und die Zahl der Stationen von 3 bis 10 variierten. Die

Mit dem Vorschlag von /NEUS 81/ (Abschätzung von bedingten Randwahrscheinlichkeiten mit Gl.(A1.25)) ergaben sich relative Fehlerabweichungen von über 30%. Die Definition der Fehlertoleranz beinhaltet, daß der absolute Fehler in Relation zur Gesamtzahl der Aufträge bzw. zur Zirkulationszeit gesetzt wird. Daher liegen die Fehlertoleranzen immer unterhalb der relativen Fehlerabweichungen und sind mit diesen in keiner Weise zu vergleichen. So kann es dann geschehen, daß die Fehlertoleranz bei einer relativen Fehlerabweichung von 30% nur 1.7% und weniger beträgt, je nachdem, wieviel Aufträge sich im Netzwerk befinden.

Aufgrund der neuen Schätzung der Randwahrscheinlichkeiten ist es nun gelungen, die durch SCAT produzierten Ergebnisse bei allen Netzwerken, bei denen sich Fehlerabweichungen von über 30% ergaben, die Abweichung von den exakten Resultaten auf weniger als 4% zu senken.

A1.1.3. REDUKTIONSMETHODE

/CHAN 75a/ haben Norton's Theorem aus der elektrischen Netzwerktheorie auf Warteschlangen angewandt und gezeigt, daß für Netze mit Produktform diese Vorgehensweise exakte Lösungen liefert. Die Idee der Parametrischen Analyse für Warteschlangennetze kann man wie folgt zusammenfassen:

Wir betrachten ein geschlossenes Warteschlangennetz mit N Bedienstationen, K Aufträgen und exponentiell verteilten Bedienzeiten.

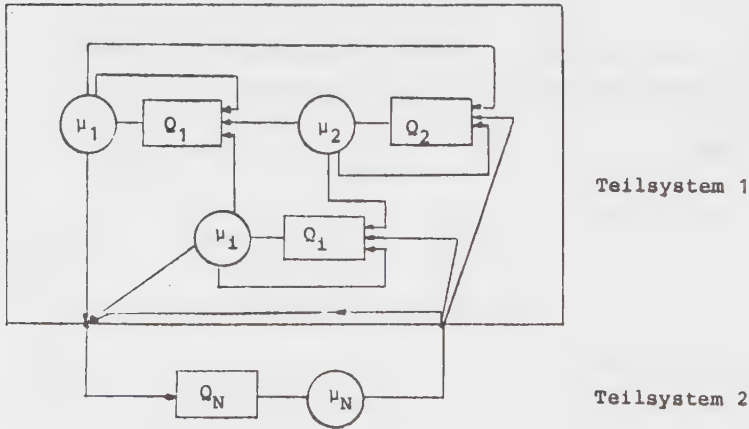


Fig. A1.3 Ein geschlossenes Warteschlangennetz

Zu diesem Netzwerk können wir ein äquivalentes Netzwerk konstruieren, indem wir eine Bedienstation auswählen (in Fig. A1.3 Bedienstation N) und die restlichen Bedienstationen zu einer einzigen Bedienstation zusammenfassen (Fig. A1.4).

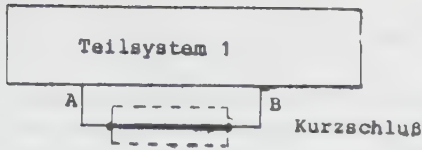


Fig. A1.4 Das kurzgeschlossene Modell

Die Bedienraten $\mu_c(k)$ für $k=1, \dots, K$ des zusammengefaßten Teilsystems erhalten wir durch "Kurzschließen" von Teilsystem 2 (in Fig. A1.5 Bedienstation N) im ursprünglich gegebenen Netz, d.h., die mittlere Bedienzeit in der Bedienstation N wird Null gesetzt (Fig. A1.5). Wir berechnen den Durchsatz durch diesen Kurzschluß in Abhängigkeit von der Anzahl der Aufträge $k=1, \dots, K$.

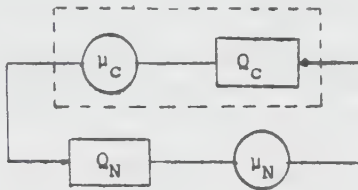


Fig. A1.5 Das resultierende System

Die Bedienrate $\mu_c(k)$ des zusammengefaßten Teilsystems ist gleich diesem Durchsatz durch den Kurzschluß. Das äquivalente reduzierte Netzwerk (Fig. A1.4) kann jetzt leicht analysiert und insbesondere die Normalisierungskonstante bestimmt werden, die die Berechnung aller interessierenden Leistungsgrößen ermöglicht.

Kritik zur Parametrischen Analyse

Betrachtet wird ein Netzwerk mit N Bedienstationen und einer Klasse mit K Aufträgen. Dann besteht die zusammengesetzte Station c aus $(N-1)$ Bedienstationen. Um den Durchsatz durch diese Station bei kurzgeschlossenener Bedienstation N zu berechnen, müssen wir also ein Netzwerk mit $(N-1)$ Bedienstationen analysieren. Geschieht das wiederum durch die Parametrische Analyse, so wird das Verfahren schließlich identisch mit dem CHW-Algorithmus von

/CHAN 75a/, bei dem $(N-1)$ -mal zwei $(K+1)$ -dimensionale Vektoren gefaltet werden müssen, was einen beträchtlichen Rechenaufwand darstellt.

Verwenden wir zur Analyse der zusammengesetzten Station irgendeinen anderen Algorithmus, so hätten wir ohne allzugroßen Mehraufwand auch gleich die Leistungsgrößen des gesamten Netzwerks mit diesem Algorithmus bestimmen können. Hinzu kommt, daß man als Ergebnis der Parametrischen Analyse lediglich die Normalisierungskonstante erhält. Um aus dieser Hilfsgröße die eigentlichen Leistungsgrößen zu gewinnen, bedarf es weiterer Berechnungen, eine Schwäche, die auch allen Erweiterungen der Parametrischen Analyse anhängt.

Erweiterungen der Parametrischen Analyse

/Krit 82/ haben die vorher beschriebene Arbeit /CHAN 75a/ auf geschlossene, offene und gemischte Produktformnetze mit mehreren Auftragsklassen erweitert. Der gemeinsame Beweis aller ihrer Erweiterungen kommt einer Wiederholung des in /CHAN 75a/ geführten Beweises in allgemeiner Form gleich.

Sie unterteilen das Gesamtnetz in zwei Teilnetzwerke, die mehr als eine Bedienstation enthalten dürfen. Im Gegensatz zur Reduktionsmethode fassen /KRIT 82/ nur eines der beiden Teilnetzwerke zu einer Bedienstation zusammen. Sie berechnen den Durchsatz durch die zusammengesetzte Station über die Zustandswahrscheinlichkeiten, was sehr umständlich und dementsprechend aufwendig ist.

/BAIS 82/ untersuchen nur einen Teil des gegebenen Netzwerks. Demzufolge teilen sie das Netzwerk in zwei Teilnetzwerke, die nun allerdings mehrere Bedienstationen umfassen dürfen. Das erste Teilnetzwerk enthält diejenige Bedienstationen, deren Leistungsgrößen bestimmt werden sollen, während das zweite Teilnetzwerk aus den nicht interessierenden Stationen besteht.

Jene Stationen werden nacheinander durch Kurzschluß eliminiert, d.h. ihre Bedienzeiten werden auf Null gesetzt. Da eine Station damit de facto verschwunden ist, wird dann eine neue Matrix der Übergangswahrscheinlichkeiten berechnet, in der diese Station dann nicht mehr auftaucht. Dieser Vorgang muß bei jeder Eliminierung einer Station wiederholt werden. Insbesondere bei großen Netzwerken ist der Aufwand dafür erheblich.

Das Konzept der Reduktionsmethode

Die Idee von /CHAN 75a/ ist, daß man eine Bedienstation kurzschließt, die restlichen Bedienstationen analysiert und zum Schluß das Gesamtnetz auf nur 2 Bedienstationen reduziert. Durch die Analyse dieses reduzierten Netzwerks erhält man dann Leistungsgrößen für das ursprünglich gegebene Netz. Der Kurzschluß einer einzigen Bedienstation bringt keine Vorteile bzgl. der Reduzierung des Rechenaufwands.

Man kann diese Idee erweitern, indem man das Gesamtnetz beliebig in 2 Teilnetzwerke zerlegt (Fig. A1.6).

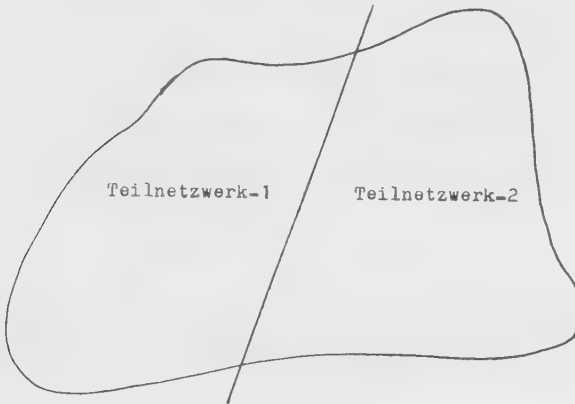


Fig. A1.6 Ein geschlossenes Warteschlangennetz

Wir können die Analyse dieses Netzes in 3 Schritten durchführen.

1. Schritt

Teilsystem 1 kurzschließen

d.h., die Bedienzeiten aller Bedienstationen im Teilnetz 1 auf Null setzen.

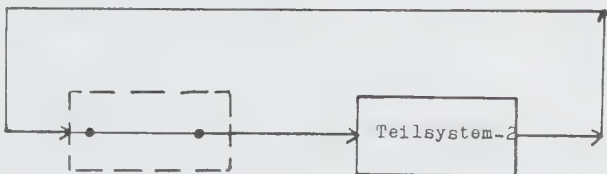


Fig. A1.7 Teilsystem 1 kurzgeschlossen

Dieses Netz untersuchen wir mit dem TUPAI-Algorithmus (Gl. A1.1) (oder auch mit einer herkömmlichen Methode) und berechnen die Durchsatzgrößen $\lambda(k)$ für $k=1, \dots, K$. Dadurch erhalten wir von den Bedienstationen in Teilsystem 2 eine zusammengefaßte Bedienstation mit den lastabhängigen Bedienraten $\mu_{c2}(k) = \lambda(k)$ für alle $k=1, \dots, K$

2. Schritt

Teilsystem 2 kurzschließen

d.h., die Bedienzeiten aller Bedienstationen im Teilsystem 2 auf 0 setzen.

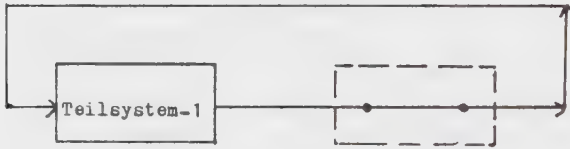


Fig. A1.8 Teilsystem 2 kurzgeschlossen

Analog erhalten wir von den Bedienstationen im Teilsystem 1 eine zusammengefaßte Bedienstation mit den lastabhängigen Bedienraten $\mu_{c1}(k) = \lambda(k)$ für alle $k=1, \dots, K$.

Bemerkung

Bei den Schritten 1 und 2 (d.h. beim Kurzschließen der Stationen) bleibt die Matrix der Übergangswahrscheinlichkeiten unverändert. Die kurzgeschlossenen Stationen degenerieren also zu einer Art Weiche. Da die Übergangswahrscheinlichkeiten aber ausschließlich durch die relativen Besuchshäufigkeiten e_i in den Algorithmus eingehen, verdient diese Tatsache bei unseren Berechnungen keine besondere Beachtung. Die e_i ($i=1, \dots, N$) bleiben unverändert. Das Kurzschließen erlaubt es uns, sogar Bedienstationen, die im Gesamtnetz keine direkte Verbindung miteinander haben, zu einem Teilnetzwerk zusammenzufassen. Bei der Analyse dieses Teilnetzwerks sind die Stationen dann durch die Kurzschlüsse der nicht im entsprechenden Teilnetzwerk befindlichen

Bedienstationen direkt miteinander verbunden.

3. Schritt

Wir gewinnen folgende Darstellung des Gesamtnetzes:

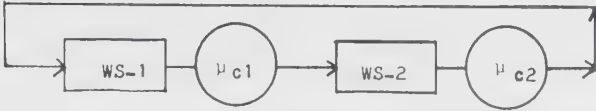


Fig. A1.9 Das resultierende System

Die erste Bedienstation stellt alle Bedienstationen des Teilsystems 1 dar, wobei die Bedienraten $\mu_{c1}(k)$ in Abhängigkeit von der Anzahl der Aufträge im 2. Schritt bestimmt wurden.

Die zweite Bedienstation stellt alle Bedienstationen des Teilsystems 2 dar, deren Bedienrate im 1. Schritt ermittelt wurden.

Das resultierende System (Fig. A1.9) kann mit Hilfe des folgenden Theorems analysiert werden.

Theorem

Für das auf 2 Stationen reduzierte Warteschlangennetz ergibt sich der Durchsatz aus der folgenden Formel:

$$\lambda(K) = \frac{\sum_{l=0}^{K-1} x(l) \quad y(K-l-1)}{\sum_{l=0}^K x(l) \quad y(K-l)} \quad (A1.35)$$

wobei

$$x(l) = \left[\begin{array}{c} 1 \\ \prod_{j=1}^l \mu_{c1}^{(j)} \end{array} \right]^{-1}$$

$$y(l) = \left[\begin{array}{c} 1 \\ \prod_{j=1}^l \mu_{c2}^{(j)} \end{array} \right]^{-1}$$

mit $x(0) = y(0) = 1$

Die anderen Leistungsgrößen für alle Stationen können anhand der ermittelten Durchsatzgrößen unter Benutzung der entsprechenden Formeln wie z.B. Gl.(A1.15) bzw. Gl.(A1.16) berechnet werden.

Beweis des Konzeptes der Reduktionsmethode

Wie wir zu Beginn des Kapitels 4 gesehen haben, gilt für die Zustandswahrscheinlichkeiten eines Netzes wegen der Produktform die folgende Formel (vgl. Gl.(A1.1)):

$$p(\underline{k}) = \frac{1}{G(K)} \prod_{i=1}^N x_i(k_i)$$

/CHAN 75a/ nennen x_i "eine Funktion von k_i ".

Für eine kurzgeschlossene Station i gilt $x_i(k_i) = 1$ für alle $i=1, \dots, N$. Schließen wir alle Stationen bis auf Station i kurz, so gibt es nur einen möglichen Zustand für jedes k und es gilt $x_j(k_j) = 1$ für alle $j = i$.

Damit gilt:

$$\begin{aligned} 1 &= \frac{1}{G(K)} x_i(k_i) \\ &= \frac{1}{G(K)} x_i(k) \text{ für } k=1, \dots, K \end{aligned}$$

Da das Netzwerk nur aus Station i besteht, gilt:

$$G_i(k) = x_i(k) = G_i(k)$$

wobei G_i die Normalisierungskonstante der i -ten Station ist.

Der Produktformansatz lautet also:

$$p(\underline{k}) = \frac{1}{G(K)} \prod_{i=1}^N G_i(k_i)$$

Die Summe der Wahrscheinlichkeiten aller erreichbaren Zustände ist

$$\sum_{i=1}^N \sum_{k_i=k} \frac{1}{G(K)} \prod_{i=1}^N G_i(k_i) = 1$$

und damit

$$G(k) = \sum_{i=1}^N \sum_{k_i=k} \prod_{i=1}^N G_i(k_i)$$

Die Normalisierungskonstante läßt sich nun auch für die 2 Teilnetzwerke ausdrücken:

$$G(k) = \sum_{i=1}^2 \sum_{k_{ci}=K} \prod_{j=1}^2 G_{cj}(k_{cj})$$

oder auch

$$G(K) = \sum_{i=1}^2 \sum_{k_{ci}=K} \prod_{j=1}^2 \sum_{s=1}^{n_{cj}} \prod_{i \in \text{TNW}j} G_i(l_{cj;i})$$

wobei k_{cij} die Anzahl der Aufträge im Teilnetzwerk (TNW) j,
 $l_{cj,i}$ die Anzahl der Aufträge in der i-ten Station des
j-ten Teilnetzwerks,
 n_{cj} die Gesamtzahl der Stationen im Teilnetzwerk j
ist.

G erhalten wir auch, indem wir die Anzahl der Aufträge in den kurzgeschlossenen Bedienstationen auf Null setzen und das innere Produkt über alle N Bedienstationen laufen lassen.

Wir definieren deshalb:

$$G_{cj,i}(k) = \begin{cases} v_n(k) & \begin{array}{l} i \text{ in TNW } j \text{ enthalten} \\ n \text{ ist Nummer der BS } cj,i \end{array} \\ \text{falls} & \text{im Gesamtnetz} \\ G_n(0) & i \text{ nicht in TNW } j \text{ enthalten} \end{cases} \quad (A1.36)$$

Wir erhalten damit:

$$G(K) = \sum_{i=1}^2 \sum_{k_{ci}=K} \prod_{j=1}^2 \sum_{\substack{n_{cj} \\ \sum_{b=1}^N l_{cj,b}}} \prod_{i=1}^N G_{cj,i}(l_{cj,i})$$

Diese Konstruktion wird mit der Faltung wie folgt berechnet:

$$G = G_{c1} \otimes G_{c2}$$

wobei

$$G_{cj} = G_{cj,1} \otimes G_{cj,2}$$

und analog zu Gl.(A1.36):

$$G_{cj,i} = \begin{cases} G_i & i \text{ in TNW } j \text{ enthalten} \\ \text{falls} & \\ NE & i \text{ nicht in TNW } j \text{ enthalten} \end{cases}$$

wobei NE das neutrale Element der Faltung (1. Einheitsvektor) ist.

Da jede Bedienstation in genau einem Teilnetzwerk enthalten ist, wird jede Bedienstation 1 mal durch seine Normalisierungskonstante und 1 mal durch NE vertreten. Streichen wir die NE als neutrale Elemente, so bleibt uns nach Anwendung des Kommutativgesetzes:

$$G = G_1 \otimes G_2 \quad (A1.38)$$

Wenn wir die Gln.(A1.8;A1.9) berücksichtigen, die Durchsatzgrößen den zusammengefaßten Bedienraten gleichsetzen und die Faltung mit Gl.(A1.38) umformen, erhalten wir das Theorem (A1.35).

q.e.d.

Bewertung

Zur Validierung der Reduktionsmethode wurden zwei Programme auf der CDC Cyber 173 am Regionalen Rechenzentrum Erlangen (RRZE) in der Programmiersprache PASCAL implementiert. Das erste Programm dient der Validierung anhand von Netzwerken mit einer Auftragsklasse. Das Konzept der Reduktionsmethode wird mit dem Algorithmus TUPAL (Gl.(A1.3)) realisiert. Die Vergleichswerte werden mit der Mittelwertanalyse, angewandt auf das Gesamtnetzwerk, ermittelt. Bei der Analyse von Netzwerken mit nur einer Auftragsklasse bestehen praktisch keine Beschränkungen hinsichtlich der Netzwerkgröße und der Auftragszahl. Auch bei extrem vielen Aufträgen im System waren keinerlei Abweichungen zwischen den Ergebnissen der Reduktionsmethode und denen der Mittelwertanalyse feststellbar.

Das zweite Programm analysiert Netzwerke mit Aufträgen aus mehreren Klassen. Die Ergebnisse wurden mit den von der Mittelwertanalyse errechneten Werten verglichen und keinerlei Abweichungen festgestellt.

Mit Hilfe des Konzepts der Reduktionsmethode können wir die Nachteile bereits bestehender Algorithmen wie z.B. MVA beseitigen. Der wesentliche Nachteil der Mittelwertanalyse für Netzwerke mit Aufträgen aus mehreren Klassen ist ihr extrem hoher Speicherplatzbedarf. So benötigen wir für die Analyse eines Netzwerks mit N Bedienstationen und K Aufträgen in R Klassen

$$O \left[N \cdot \prod_{r=1}^R (K_r + 1) \right]$$

Speicherplätze nur zur Speicherung der $\bar{k}_i(k)$ - Werte (vgl. Gl.(A1.22)).

Hinzu kommt noch die Speicherung der bedingten Wahrscheinlichkeiten $p_i(l/k)$ (vgl. Gl.(A1.18) bzw. (A1.19)), die von der Anzahl der lastabhängigen Stationen mit $m_i > 1$ und vom maximalen m_i dieser Stationen abhängt.

Der Speicherplatzbedarf kann durch die Reduktionsmethode verkleinert werden auf

$$\left(\max_{j=1,2} \{N_{Cj}\} + 2 \right) \prod_{r=1}^R (K_r + 1)$$

wobei N die Anzahl der Bedienstationen des Teilnetzwerks j ist.

Der Summand 2 ist der Bedarf für die Durchsatzgröße des gerade mit der Mittelwertanalyse untersuchten Teilnetzwerks und für die bisher mit der Faltung ermittelte Durchsatzgröße (Gl.(A1.35)).

Diese Ersparnis kann nur bei dynamischer Speicherplatzbelegung erreicht werden, d.h. bei Anmeldung und Belegung von benötigtem und unmittelbarer Freigabe von nicht mehr benötigtem Speicherplatz während des Programmlaufs. Da wir das Netzwerk in N Teilnetzwerke mit je einer Bedienstation teilen können, kommen wir im Minimum mit

$$3 \cdot \prod_{r=1}^R (K_r + 1)$$

Speicherplätzen aus. Das entspräche allerdings dem CHW-Algorithmus /CHAN 75a/. Dieser Algorithmus verbraucht jedoch, wie auch die Faltung der Teilnetzwerke, sehr viel Rechenzeit.

Im Hinblick auf den Betriebsmittelbedarf gibt die Reduktionsmethode uns damit eine große Flexibilität zwischen den beiden Extrema Mittelwertanalyse und CHW-Algorithmus.

viel Speicherplatz
wenig Rechenzeit

wenig Speicherplatz
viel Rechenzeit



Extrem speicherplatzintensiv ist die Speicherung der bedingten Wahrscheinlichkeiten im Falle der lastabhängigen Stationen mit $m_i > 1$ in der MVA. Für ihre Speicherung benötigen wir

$$N \cdot \left(\max_{i=1, \dots, N} \{m_i\} - 1 \right) \prod_{r=1}^R (K_r + 1)$$

Speicherplätze. $\text{MAX} \{m_i\}$ ist die maximale Anzahl der Bedieneinheiten, die eine Typ-1-Bedienstation aufweist. Der Speicherplatzbedarf der $p_i(l/k)$ beträgt also das $(\text{MAX} \{m_i\} - 1)$ -fache des Bedarfs der $\bar{K}_i(k)$.

Dieser Bedarf kann radikal reduziert werden, wenn wir in jedem Teilnetzwerk höchstens eine (oder auch zwei) Bedienstation(en) vom Typ 1 mit $m_i > 1$ zulassen. Im erstem Fall würde sich der Speicheraufwand auf

$$\left(\max_{i=1, \dots, N} \{m_i\} - 1 \right) \prod_{r=1}^R (K_r + 1)$$

reduzieren. Da die maximale Anzahl der Bedieneinheiten meist sehr viel kleiner ist als die Anzahl der Bedienstationen im Gesamtnetzwerk, ist die Speicherplatzersparnis enorm. Zudem wird der reservierte Speicherbereich besser genutzt, da i.d.R. nur ein Teil der Bedienstationen vom Typ 1 mit $m_1 > 1$ ist. Angesichts dessen fällt der zusätzliche Rechenaufwand für die Faltung, mit dem diese Ersparnis erkauft werden muß, nicht so sehr ins Gewicht. Dies zumal dann, wenn man zum Vergleich etwa die Laufzeit von Simulationen heranzieht.

Der Aspekt der Speicherplatzersparnis fällt bei der Anwendung auf modernen Großrechenanlagen mit ihren virtuellen Speichern kaum ins Gewicht. Hier ist höchstens die Erhöhung der Rechenzeit bei häufigen Ein- und Auslagerungen erwähnenswert. Bei der Implementation der algorithmisch einfachen analytischen Methoden auf Kleinrechnern ist jedoch eine flexible Anpassung des Speicherplatzbedarfs an die beschränkten Ressourcen eminent wichtig. Da diese Anwendung, auch als Folge des Kleinrechner-Booms, sehr stark an Bedeutung gewonnen hat, stellt die Reduktionsmethode auf diesem Gebiet eine attraktive Ergänzung der bisherigen Methoden dar.

A1.2. NICHTPRODUKTFORM-NETZWERKE

Wenn ein Warteschlangennetz die in Abschnitt A1.1. eingeführten Bedingungen für eine Produktformlösung nicht erfüllt, können wir die im vorhergehenden Abschnitt behandelten Algorithmen nicht anwenden. Wenn ein Netz z.B. eine Station mit FCFS-Strategie und nicht-exponentieller Bedienzeitverteilung enthält, was in vielen realen Systemen der Fall ist, erhalten wir keine Produktformlösung.

Es gibt 3 Möglichkeiten, um Lösungen für Nichtproduktform-Netze zu bekommen:

- i) Wir verschärfen die Annahmen dahingehend, daß wir ein Produktformnetz erhalten, das das reale System approximiert. Dieses System untersuchen wir dann mit einem exakten Algorithmus.
- ii) Wir verwenden die numerischen Methoden /STEW 78, HERZ 75/, die über die Übergangsratenmatrix zunächst die Zustandswahrscheinlichkeiten und daraus die Leistungsgrößen des Netzes bestimmen. Die numerischen Methoden sind wegen der aufwendigen Übergangsratenmatrix nur auf sehr kleine Netze anwendbar. Für größere Netze wird diese Matrix sehr groß und nicht mehr speicherungsfähig.
- iii) Wir lösen das Modell durch Anwendung von Approximationsmethoden.

Es existieren mittlerweile zahlreiche approximative Algorithmen für Nichtproduktform-Netzwerke, von denen einige sehr gute Ergebnisse liefern. Die wichtigsten Approximationsmethoden unterteile ich wie folgt in zwei Klassen:

- 1) Produktform-Approximationen
 - a) Diffusions-Approximation /KOBA 74/
 - b) Erweiterte Produktform-Methode (EPF) /SHUM 76/

2) Aggregations- / Dekompositionsverfahren

- a) Iterative Approximation /CHAN 75b/
- b) Dekompositionsapproximation /COUR 77/
- c) Methode von Kühn /KUHN 79/
- d) Methode von Marie /MARI 80/

Diese Methoden werden in Abschnitt 1.2 kurz erläutert. Die Einzelheiten können entweder in der Originalliteratur oder auch in /BOAK 82/ nachgelesen werden.

A1.2.1. ERWEITERTE MITTELWERTANALYSE

Die in diesem Abschnitt betrachteten Warteschlangennetze bestehen ausschließlich aus Stationen mit nur einer Bedieneinheit ($m_i=1$). Die Ankunftszeiten der Aufträge sind exponentiell verteilt. Die Aufträge werden in jeder Station nach FCFS abgearbeitet und die Bedienzeiten in den Stationen sind nicht exponentiell, sondern erlang- oder hyperexponentiell verteilt mit 2 Phasen.

In einem geschlossenen Netzwerk mit M/G/1-FCFS Stationstypen ergibt sich die mittlere Verweilzeit der Aufträge der r-ten Klasse in der i-ten Station aus der folgenden Formel:

$$\bar{t}_{ir}(\underline{k}) = \frac{1}{\mu_{ir}} \left[1 + \sum_{s=1}^R \bar{k}_{is}(\underline{k}-1_r) + 0.5 \cdot x_{ir} \cdot \lambda_r(\underline{k}-1_r) (c_{ir}^2 - 1) \right] \quad (A1.39)$$

wobei $r=1, \dots, R$ die Auftragsklasse,
 $\underline{k}=(k_1, \dots, k_R)$ ein Zustandsvektor,
 $(\underline{k}-1_r)=(k_1, \dots, k_r-1, \dots, k_R)$ ein Auftrag weniger in Klasse r,
 c_{ir} der Variationskoeffizient der Bedienzeitverteilung in der i-ten Station und der r-ten Auftragsklasse.

Als Initialwerte benutzen wir:

$$\bar{t}_{ir}(b_r) = x_{ir} \quad \lambda_r(b_r) = \frac{1}{\sum_{i=1}^N x_{ir}}$$

b ist Einheitsvektor

Ableitung

Da die Ableitung für den Mehrklassenfall aufwendig ist, gehe ich von einer Auftragsklasse aus. Der Ausgangspunkt für den Beweis ist die folgende Hilfsfunktion, die von /SAUE 81/ als eine Erweiterung der LBANC-Methode vorgeschlagen wurde:

$$q_i(k) = x_i \left[G(k-1) + q_i(k-1) + 0.5 x_i G(k-2)(c_i-1) \right] \quad (A1.40)$$

mit $q_i(0) = 0$; $G(0) = 1$; $G(-y) = 0$ für alle $y \in N$.

Nach /SAUE 81/ gilt:

$$q_i(k) = \bar{k}_i(k) * G(k) \quad (A1.41)$$

Wir substituieren Gl.(A1.41) in Gl.(A1.40):

$$\bar{k}_i(k) * G(k) = x_i \left[G(k-1) + \bar{k}_i(k-1) G(k-1) + 0.5 x_i G(k-2)(c_i-1) \right]$$

Daraus erhalten wir

$$\bar{k}_i(k) = x_i \left[\frac{G(k-1)}{G(k)} + \bar{k}_i(k-1) \frac{G(k-1)}{G(k)} + 0.5 x_i \frac{G(k-2)}{G(k)} c_i^2 - 1 \right]$$

Wir erweitern den dritten Summanden um $G(k-1)$:

$$\bar{k}_i(k) = x_i \left[\frac{G(k-1)}{G(k)} + \bar{k}_i(k-1) \frac{G(k-1)}{G(k)} + 0.5 x_i \frac{G(k-2)}{G(k)} \cdot \frac{G(k-1)}{G(k-1)} \cdot (c_i^2 - 1) \right]$$

klammern $\frac{G(k-1)}{G(k)}$ aus und berücksichtigen, daß

$$\bar{k}_i(k) = x_i \lambda(k) \left[1 + \bar{k}_i(k-1) + 0.5 x_i \lambda(k-1)(c_i-1) \right]$$

Mit Littles Gesetz $\bar{k}_i(k) = \lambda(k) * e_i * \bar{t}_i(k)$
und $\lambda_i(k) = e_i * \lambda(k)$ erhalten wir

$$\bar{t}_i(k) = \frac{1}{\mu_i} \left[1 + \bar{k}_i(k-1) + 0.5 x_i \lambda(k-1)(c_i-1) \right]$$

Analog kann man die Ableitung für den Mehrklassenfall durchführen.

Da Littles Gesetz für alle Stationstypen gilt, können wir die Formeln für den Durchsatz (Gl.(A1.20)) und die mittlere Anzahl von Aufträgen (Gl.(A1.22)) benutzen, die aus Littles Gesetz abgeleitet wurden.

A1.2.2. SIMVAG (Short Iterative Mean Value Analysis for General Networks)

Die EMVA hat das gleiche Problem wie die MVA, daß sie eine komplette Lösung für alle Auftragszahlen von 0 bis K erfordert. Damit entstehen auch Speicherkapazitätsprobleme, wenn Mehrklassennetze analysiert werden sollen. Wie bei /SCHW 79/ können wir die Iteration der Methode nicht von 0 an, sondern mit einem beliebig angenommenen Initialwert starten.

Der Algorithmus läuft folgendermaßen ab:

1. Schritt Ermittlung von Initialwerten

- Berechne die mittlere Anzahl von Aufträgen aus /SCHW 79/

$$\bar{k}_i = \frac{K}{N}$$

- Berechne den Durchsatz aus

$$\lambda = \frac{K}{N \sum_{i=1}^N x_i + (K-1) x_b} \quad (A1.42)$$

wobei

$$x_b = \frac{\sum_{i=1}^N x_i}{N}$$

Ableitung von λ :

Nach der MVA gilt:

$$\lambda = \frac{K}{N \sum_{i=1}^N x_i [1 + \bar{k}_i (K-1)]}$$

Nach /SCHW 79/ gilt:

$$\bar{k}_i (K-1) = \frac{K-1}{K} \bar{k}_i$$

mit dem Anfangswert $\bar{k}_i = \frac{K}{N}$

Damit erhalten wir

$$\lambda = \frac{K}{N \sum_{i=1}^N x_i + (K-1) \cdot \frac{\sum_{i=1}^N x_i}{N}}$$

woraus dann Gl.(A1.42) entsteht.

q.e.d.

2. Schritt

Ermittlung der Leistungsgrößen

- Berechne die mittlere Verweilzeit mit Gl.(A1.39), den Durchsatz mit Gl.(A1.20) und die mittlere Anzahl von Aufträgen mit Gl.(A1.22).

3. Schritt

Terminationstest

- Überprüfe als Abbruchkriterium, ob sich der Wert für $\bar{K}_i$ dieser Iteration von dem der vorhergehenden Iteration um mehr als ϵ unterscheidet. Im Falle der Überschreitung muß der Algorithmus ab 2. Schritt wiederholt werden. Sonst erhält man die Leistungsgrößen.

Um gute Ergebnisse zu erzielen, sollte nach den bisherigen Erfahrungen ϵ zwischen 0.01 und 0.1 liegen.

Bewertung

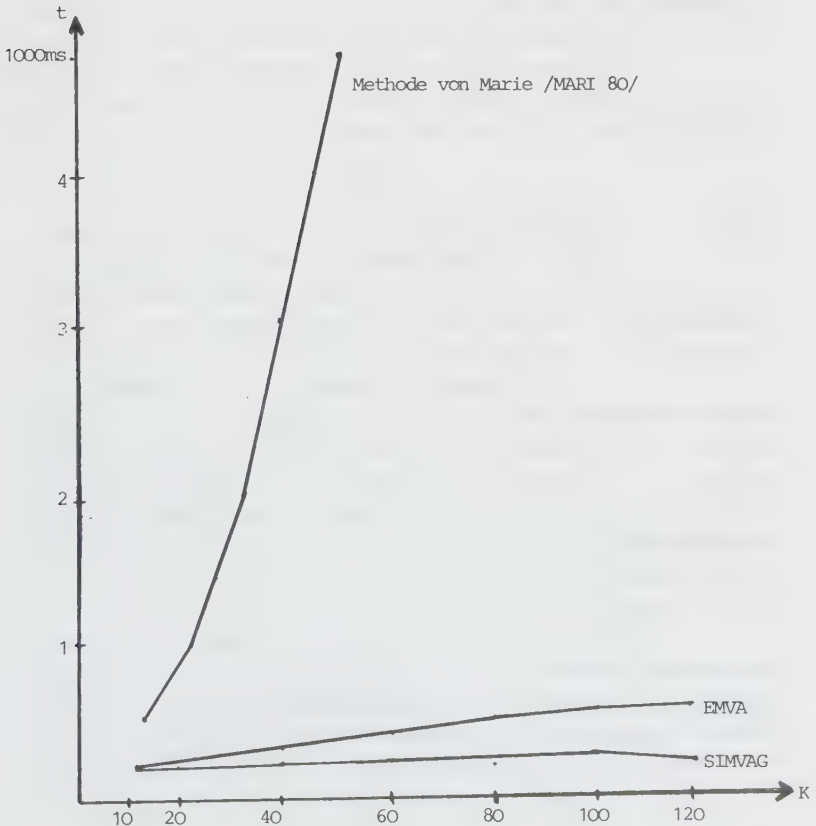
Die EMVA und SIMVAG wurden in der Programmiersprache PASCAL auf der Rechenanlage VAX 11/780 implementiert. Es wurden ungefähr 25 verschiedene Warteschlangennetze mit verschiedener Anzahl von Aufträgen und Stationen analysiert. Die Ergebnisse wurden mit den Ergebnissen der Methode von /MARI 80/, der EPF-Technik /SHUM 76/ und der Simulation verglichen.

Die Simulation wurde mit Hilfe des Simulationspaketes GPSS-Fortran Version 3 /SCHM 84/ durchgeführt, ein Programmpaket, das am IMMD IV entwickelt wurde und auf der Rechenanlage TR 440 implementiert war.

In den Fällen, in denen ein Netz ausschließlich Stationen mit

Erlang-2-Bedienzeitverteilung enthält, habe ich sehr gute, von den Simulationswerten nur um 1-5% abweichende Ergebnisse erhalten. Im Falle der hyperexponentiellen Bedienzeitverteilung sind die Abweichungen mit 7-20% etwas größer. Das bedeutet, daß EMVA bzw. SIMVAG für Erlang-2-Bedienzeitverteilungen zu empfehlen sind. Die EMVA hat sehr gute Rechenzeiten im Vergleich zur EPF-Methode und zur Methode von /MARI 80/. Die Ergebnisse der SIMVAG unterscheiden sich bei $\varepsilon = 0.05$ von denen der anderen Methoden nur unwesentlich. Dieser Wert für ξ hat sich heuristisch als sehr gut erwiesen.

Die folgende Fig.A1.10 macht das Verhältnis dieser Werte deutlich.



ANHANG II

A2. SIMULATION DES GESAMTSYSTEMS

Um das Gesamtmodell auf seine Richtigkeit hin überprüfen zu können, wurde es in seiner dargestellten Struktur, Systemmodell Kap. 2, mit dem GPSS-Fortran-Simulator /SCHM 84/, Version 3, am Regionalen Rechenzentrum Erlangen (RRZE) auf dem Rechner Telefunken TR440 simuliert.

Um das Gesamtmodell auf dem Simulator übertragen zu können, müssen zunächst die enthaltenen Bedienstationen auf GPSS-F-Stationstypen abgebildet werden. Für das Modell eignen sich die sogenannten Multifacilities sowie die Pools, wobei eine Multifacility eine Bedienstation ist, die aus mehreren Bedieneinheiten sowie einer gemeinsamen Warteschlange besteht. Die Prozessoren werden also im Simulator durch die Multifacilities modelliert. Die Prozessorarbeitungsstrategien sind wie im Systemmodell, Kap. 2, erwähnt PS. In der Simulation wird die Strategie FCFS auch untersucht.

Prozeßwechsel werden vorgenommen, wenn ein in Bedienung befindlicher Prozeß eine kritische Aktion (d.h. Senden und Empfangen von Nachrichten) erreicht und diese nicht ausführen kann, und (je nach Warteschlangenstrategie), wenn die Zeitscheibe eines Prozesses abgelaufen bzw. sein Zyklusende erreicht ist.

Ein Pool ist eine Bedienstation, die aus einem endlichen Puffer sowie einer Warteschlange für momentan nicht bedienbare Aufträge besteht. Solange der Pool nicht voll ist, werden ankommende Aufträge in den Puffer genommen und verlassen diesen nach einer gewissen Verzögerungszeit $t_{\text{POOL}} > 0$ oder nach Eintritt eines bestimmten Ereignisses wieder. Bei vollem Puffer werden ankommende Aufträge so lange in der Warteschlange vor dem Pool blockiert, bis ein Platz im Puffer frei wird und der anfordernde Auftrag bedient werden kann.

Um die Kommunikationswarteschlangen des Gesamtmodells durch Pools modellieren zu können, muß es möglich sein, ankommende Aufträge nicht nur dann zu blockieren, wenn der Puffer belegt ist (SEND), sondern auch dann, wenn aus leerem Puffer gelesen werden soll (WAIT). Im Simulator GPSS-Fortran ist diese Möglichkeit

durch explizites Blockieren (Unterprogramm BLOCK) gegeben. Somit können die Kommunikationswarteschlangen im Simulator durch Pools adäquat wiedergegeben werden, wenn die Warteschlangen vor den Puffern nach Strategie FCFS abgearbeitet werden.

Um nun ein vollständiges Abbild des Gesamtmodells für den Simulator zu bekommen, müssen die Übergänge zwischen den Stationen des Warteschlangenmodells in geeigneter Weise festgelegt werden. Dazu wird im Simulationsprogramm eine sogenannte Prozeßbeschreibungsmatrix PROMAT erzeugt, die Zeitpunkt und Art einer kritischen Operation eines Prozesses enthält. Gemäß diesen Angaben verläßt ein Prozeß während des Simulationslaufes die Multifacility und wandert zu einem der Pools. Die Zeitpunkte der kritischen Aktionen werden durch die Angabe einer exponentiell verteilten Rate im Eingabedatensatz des Simulationsprogramms bestimmt. Die Rate gibt an, welchen Anteil die kritischen Aktionen an den gesamten Operationen des Prozesses haben sollen.

Als Nächstes ist der Ratenvektor KRIRAT zu besetzen (Länge \$1). Hierzu wird für jeden Prozeß ein separater Wert eingelesen. Er gibt an, in welchen exponentiell verteilten Abständen kritische Operationen erfolgen sollen.

Um die einzelnen Prozesse sequentiell in der Prozeßbeschreibungsmatrix abspeichern zu können und somit den Speicherbedarf möglichst gering zu halten, werden in der Matrix KRITAN zu jedem Prozeß Startadresse und Länge in PROMAT abgespeichert. Prozeß 1 hat die Startadresse 1, Prozeß i erhält als Startadresse den nach Eintrag von Prozeß i-1 nächsten freien Speicherplatz in PROMAT. Die Anzahl der kritischen Operationen pro Prozeß wird aus der Gesamtprozeßlänge und der Rate des Auftretens kritischer Aktionen errechnet.

Damit jeder Prozeß im Gesamtsystem an der Prozeßkommunikation beteiligt ist, erscheint es sinnvoll, für jeden Prozeß zumindest die Ausführung einer kritischen Operation vorzusehen, denn im Systemmodell Kap. 2 war angenommen, daß jeder Prozeß mit jedem kommunizieren kann.

Die Art der kritischen Operationen wird durch zwei Uniformverteilungen bestimmt. Die erste legt fest, ob es sich um eine SEND- oder um eine WAIT-Operation handeln soll. Im letzteren Fall wird in der Prozeßbeschreibung der Wert 0 eingetragen. Handelt es

sich um eine SEND-Operation, so wird durch die zweite Uniformverteilung festgelegt, an welchen der Pools die Nachricht gesendet werden soll. In diesem Falle wird in der Prozeßbeschreibungsmatrix der Wert i ($1 < i < n$) eingetragen. Falls nicht ausdrücklich anders vermerkt, werden im Folgenden immer im Durchschnitt je zur Hälfte SEND- und WAIT-Operationen ausgeführt; eine SEND-Operation erfolgt im Mittel an alle anderen Prozesse mit der gleichen Wahrscheinlichkeit.

BEARBEITUNG VON PROZESSEN

Die Bearbeitung von Prozessen während eines Simulationslaufes läßt sich grundsätzlich in zwei Abschnitte untergliedern:

- Bearbeitung in unkritischen Phasen in der Multifacility
- Bearbeitung von kritischen Operationen an den Pools

Im folgenden sollen zunächst beide Abschnitte getrennt dargestellt werden.

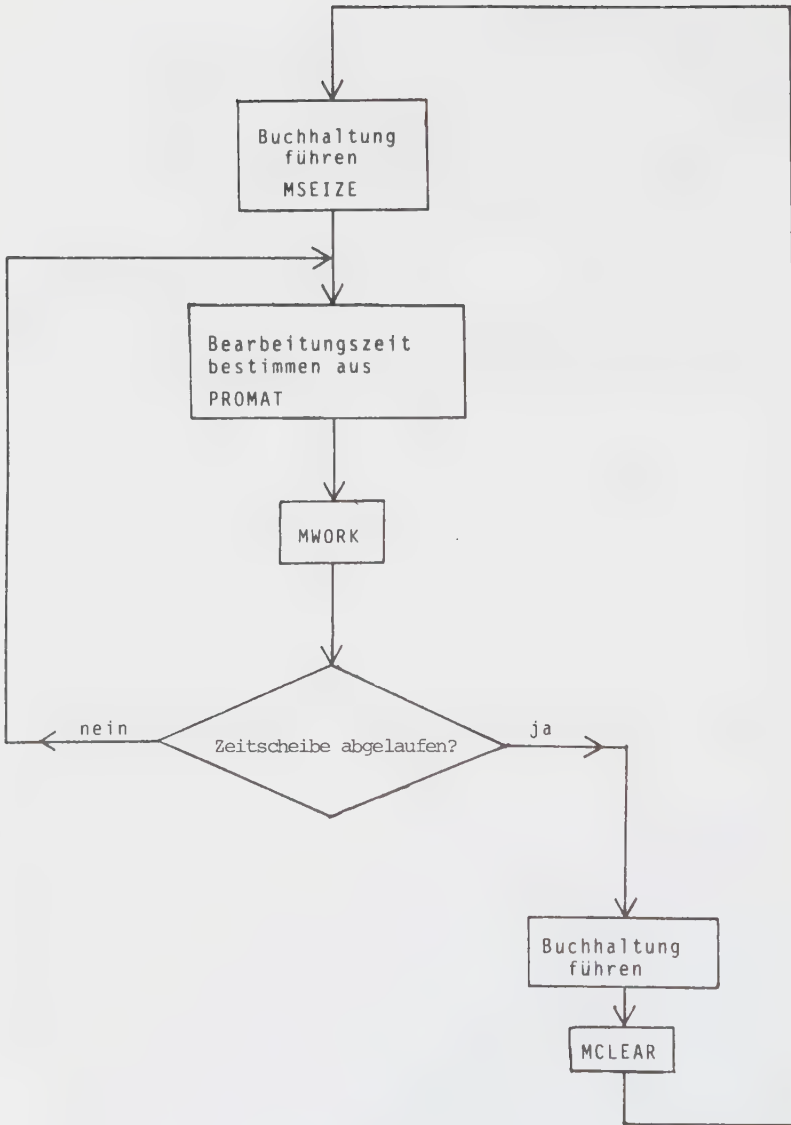
1) Bearbeitung in unkritischen Phasen

Die Bearbeitung eines Auftrags in der Multifacility umfaßt hauptsächlich drei Phasen:

Durch den Aufruf des Unterprogramms MSEIZE wird ein ankommender Prozeß in die Warteschlange vor der Multifacility eingereiht. Sind alle Bedieneinheiten belegt, so verbleibt er darin so lange, bis alle gemäß Abarbeitungsstrategie der Warteschlange vor ihm zur Bedienung anstehenden Prozesse bedient wurden oder sich in Bedienung befinden. Wird danach eine Bedieneinheit frei, so belegt der bisher auf Bedienung wartende Prozeß dieses durch Aufruf des Unterprogramms MWORK. Beim Aufruf dieses Unterprogramms wird in der Parameterliste die Dauer der Bearbeitung übergeben, die aus der Prozeßbeschreibungsmatrix entnommen wird. Die Bearbeitung von Prozessen findet in der unkritischen Phase ohne Verdrängung statt, sodaß die angegebene Bedienungszeit in jedem Fall dem anfordernden Prozeß vollständig zur Verfügung steht.

Sieht man von Prozeßwechseln durch das Auftreten kritischer Operationen einmal ab, so wird die belegte Bedieneinheit erst bei Ablauf der Zeitscheibe im Falle der PS-Abarbeitungsstrategie oder Erreichen des Zyklusendes eines Prozesses im Falle der FCFS durch das Unterprogramm MCLEAR wieder freigegeben. Der freigebende Prozeß reiht sich sodann wieder in die Warteschlange vor der Multifacility ein.

Insgesamt läßt sich also die Bearbeitung in unkritischen Phasen ohne Berücksichtigung des Auftretens von vorzeitigen Prozeßwechseln, verursacht durch die Ausführung von kritischen Operationen, wie auf der folgenden Seite abgebildet darstellen.

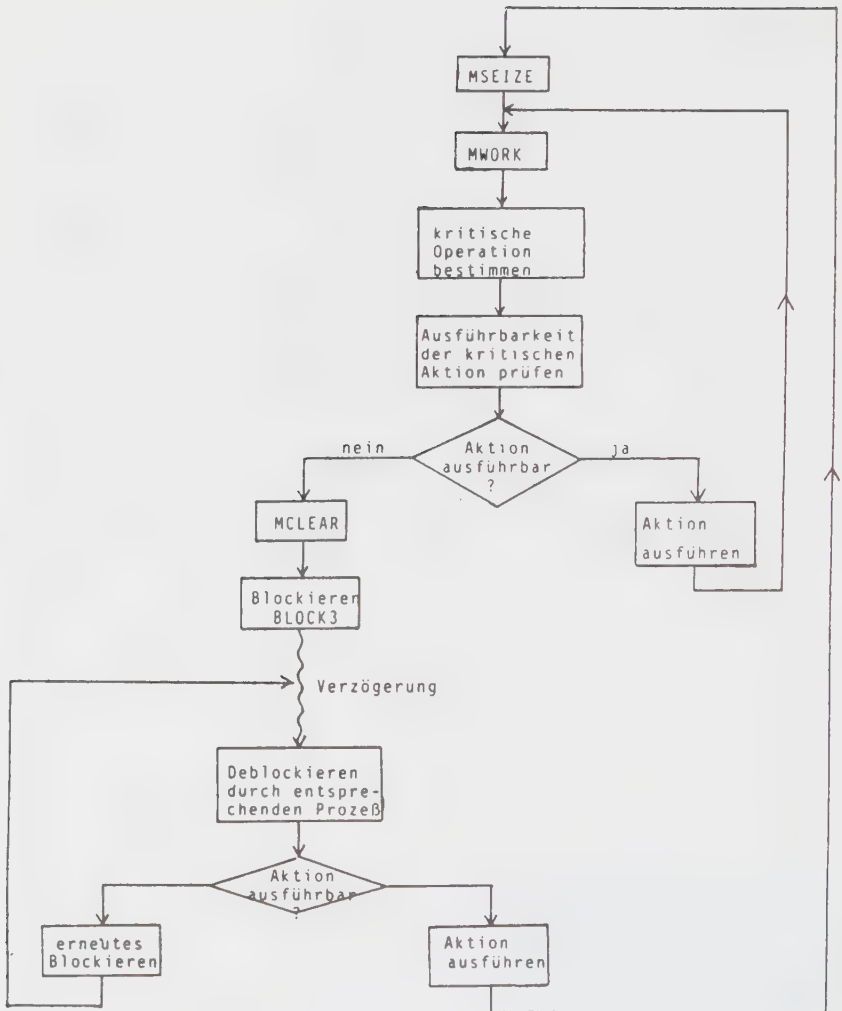


Bearbeitung in der Multifacility ohne vorzeitige
Prozeßwechsel durch kritische Operationen

2) Bearbeitung von kritischen Operationen

Eine kritische Operation liegt immer dann vor, wenn Nachrichten gesendet beziehungsweise empfangene Nachrichten verarbeitet werden sollen. Grundsätzlich wird bei beiden Arten zunächst die Ausführbarkeit der anstehenden kritischen Aktion überprüft; bei Ausführbarkeit wird nach dem Senden / Verarbeiten der Nachricht mit der Bearbeitung der nächsten unkritischen Phase dieses Prozesses fortgefahren, bei Nichtausführbarkeit wird der aufrufende Prozeß blockiert und wartet anschließend darauf, von einem anderen Prozeß deblockiert zu werden.

Eine kritische Operation ist genau dann nicht ausführbar, wenn entweder bei leerem eigenen Nachrichtenpuffer eine WAIT-Operation oder aber bei vollem Zielnachrichtenpuffer eine SEND-Operation ausgeführt werden soll. In beiden Fällen hat also der blockierte Prozeß keinen direkten Einfluß auf seine Deblockierung, sondern kann nur durch externe Prozesse eine Deblockierung erfahren. Das Geschehen bei Ausführung einer kritischen Aktion zeigt das auf der folgenden Seite abgebildete Flußdiagramm:



Ausführung einer kritischen Operation

PROTOKOLLIERUNG UND AUSWERTUNGEN

Zur Gewinnung der gesuchten Leistungsgrößen werden während eines Simulationslaufes zahlreiche Protokollierungsvariablen laufend auf dem aktuellen Stand gehalten und am Ende der Simulation ausgewertet; zum Teil sind dies vom Simulator bereitgestellte Protokollierungsvariablen, die BINs, zum anderen Teil benutzerzugängliche reservierte Felder in der Transactionmatrix sowie benutzerdefinierte Größen.

Nachstehend ist der Zweck der einzelnen Größen sowie die Bestimmung der Leistungsgrößen daraus näher erläutert.

1) BINs

BINs sind vom System bereitgestellte Variablen, die zu Ende eines Simulationslaufes durch systeminterne Auswertungsroutinen bearbeitet werden können. Im vorliegenden Programm wird in den BINs mit Nummer i ($1 \leq i \leq 49$) festgehalten, welche Zeit der i -te Prozeß während des Simulationslaufes an einem Pool blockiert war, und daraus später die Blockierzeit pro Zyklus bestimmt.

In BIN 50 wird protokolliert, wieviele Aufträge sich momentan in der Multifacility befinden, sowie deren Verweildauer. Nach Aufruf des Unterprogramms ENDBIN stehen dem Benutzer dann die mittlere Anzahl von Aufträgen und die mittlere Verweilzeit in der Multifacility zu Verfügung.

In BIN 51 schließlich wird vermerkt, welche Zeit eines oder mehrere der Bedieneinheiten leerstanden, um daraus später die Auslastung der Multifacility zu bestimmen.

2) Transactionmatrix und benutzerdefinierte Größen

Die Transactionmatrix enthält alle für einen Prozeß wichtigen Informationen. Darüber hinaus stehen dem Benutzer die letzten Felder der Matrix als Leerfelder zur Verfügung, in denen benutzereigene Werte eingetragen werden können. Diese sind im vorliegenden Fall folgendermaßen belegt:

Feld-Nr.	Inhalt
10	Zähler für aktuellen Zyklusstand
11	Anzahl der ausgeführten kritischen Aktionen
12	Anzahl der Zyklusdurchläufe
13	Vermerk für Blockierung im Zyklus
14	Anzahl der blockierten Zyklen

Als benutzerdefinierte Größen verdienen die logischen Variablen BLOZYK, WAIT, SEND, ERFOIG und ZYKEND Erwähnung; sie dienen programmintern als wesentliche Hilfsmittel bei der Bestimmung des Weges, den ein Prozeß im Laufe seiner Bearbeitung durch das Modell nimmt.

3) Leistungsgrößen

Die Leistungsgrößen des simulierten Modells lassen sich aus den angesprochenen Protokollierungsvariablen wie folgt berechnen:

a) Auslastung:

$$RHO = 1 - \frac{BIN(i,7)}{\$3 * TEND}$$

b) Durchsatz:

$$LAMBDA = \$3 * RHO * 0,02$$

c) Mittlere Blockierzeit des i-ten Prozesses:

$$\bar{t}_B(i) = \frac{BIN(i,7)}{TX(i,14)}$$

d) Blockwahrscheinlichkeit des i-ten Prozesses:

$$p_B(i) = \frac{TX(i,14)}{TX(i,12)} * 100$$

e) Mittlere Wartezeit des i-ten Prozesses vor der Multifacility:

$$\bar{t}_w(i) = \frac{T}{TX(i,12) + TX(i,10) / KRITAN(i,2) - PRODAU(i) + \frac{BIN(i,7)}{TX(i,14)}}$$

f) Mittlere Anzahl von Aufträgen in der Multifacility:

$$\bar{K} = BINSTA(50,2)$$

g) Mittlere Verweilzeit eines Auftrages in der Multifacility:

g1) zwischen zwei Prozeßwechseln:

$$\bar{t}_1 = BINSTA(50,1)$$

g2) zwischen Zyklusende und Wiederbearbeitung:

$$\bar{t}_2 = \frac{\sum_{i=1}^{51} \bar{t}_w(i)}{51}$$

In Formel b) wird zur Berechnung des Gesamtdurchsatzes an der Multifacility nach der Formel von LITTLE die Anzahl der Bedienelemente des Multiprozessors mit der Auslastung und der mittleren Bedienrate des Prozesses multipliziert; da alle Prozesse im Mittel 50 Zeiteinheiten lang sind und sich die Bedienrate aus dem Kehrwert der Bedienzeit ergibt, müssen hier 51 und RHO mit 0.02 multipliziert werden.

ANHANG III

A3. Ergebnisse für Flockiernetze

Beispiel 1

$N = 3$, $K = 5$, $1/\mu_1 = 1$, $1/\mu_2 = 0.0$, $1/\mu_3 = 2$
 $n_i = 2$ ($i=1,2,3$)

	SIM	ZRTM	Abw	IVARIO	Abw	PROBIO	Abw	OVAP
$\bar{t}_1$	2.921	2.926	-	2.652	10.1	3.058	4.5	1.752
$\bar{t}_2$	4.129	4.207	1.8	3.408	21.1	2.977	32.6	1.478
$\bar{t}_3$	4.217	4.504	6.8	3.367	25.5	3.420	23.3	7.212
$\bar{k}_1$	1.297	1.257	3.2	1.407	8.4	1.617	19.7	0.839
$\bar{k}_2$	1.832	1.808	1.3	1.808	1.3	1.575	16.2	0.708
$\bar{k}_3$	1.871	1.935	3.2	1.786	4.5	1.809	3.3	3.454
λ	0.444	0.430	2.3	0.530	19.5	0.529	19.1	0.479

Beispiel 2

$N = 3$, $K = 25$, $1/\mu_1 = 1$, $1/\mu_2 = 0.5$, $1/\mu_3 = 0.333$
 $n_1 = 12$, $n_2 = 10$, $n_3 = 14$

	Sim	ZRTM	Abw	IVARIO	Abw	PROBIO	Abw	OVAP
$\bar{t}_1$	11.890	12.001	1.0	11.912	0.2	8.677	37.0	23.500
$\bar{t}_2$	0.986	0.999	1.4	1.011	2.5	6.138	63.0	1.000
$\bar{t}_3$	12.230	12.002	1.8	12.022	1.7	4.041	66.9	0.500
$\bar{k}_1$	11.840	12.000	1.3	11.938	0.8	11.504	2.8	23.500
$\bar{k}_2$	0.980	0.999	1.8	1.014	3.3	6.138	87.9	1.000
$\bar{k}_3$	12.180	12.001	1.4	12.049	1.1	5.358	56.0	0.500
λ	0.995	1.000	-	1.002	0.7	1.326	24.9	1.000

SIM= Simulation

ZRTM= Zustandsraumtransformationmethode

IVARIO= Mittelwertanalyse für Flockiernetze

PROBIO= Produktformlösungen für Flockiernetze

OVAP = Ohne Berücksichtigung der Kapazitäten

Peispiel 3

$N = 3$, $K = 8$, $1/\mu_1 = 0.6$, $1/\mu_2 = 0.8$, $1/\mu_3 = 1$
 $N_1 = 6$, $N_2 = 5$, $N_3 = 8$

	Sim	ZRTH	Abw	MVARIO	Abw	PROBLO	Abw	OKAP
$\bar{t}_1$	1.490	1.445	3.0	1.272	14.6	1.319	13.0	1.272
$\bar{t}_2$	2.214	2.283	3.1	2.465	11.3	1.955	13.2	2.465
$\bar{t}_3$	4.794	4.807	-	4.798	-	5.304	9.6	4.798
$\bar{k}_1$	1.403	1.354	3.5	1.192	14.5	1.230	14.1	1.922
$\bar{k}_2$	2.086	2.140	2.5	2.311	10.7	1.823	14.4	2.311
$\bar{k}_3$	4.512	4.506	-	4.497	-	4.947	8.8	4.497
λ	0.942	0.937	-	0.937	-	0.933	1.0	0.937

Peispiel 4

$n = 4$, $K = 8$

$1/\mu_1 = 0.6$, $1/\mu_2 = 0.7$, $1/\mu_3 = 0.4$, $1/\mu_4 = 0.8$
 $l_i = 3$ (i=1,2,3,4)

	Sim	ZRTH	Abw	MVARIO	Abw	PROBLO	Abw	OKAP
$\bar{t}_1$	1.647	1.745	5.6	1.467	10.9	1.807	9.6	1.467
$\bar{t}_2$	1.865	1.963	5.0	2.121	13.7	1.961	5.1	2.121
$\bar{t}_3$	1.767	1.666	6.1	1.478	16.4	1.255	29.0	0.678
$\bar{t}_4$	2.505	2.500	-	2.264	9.6	2.071	17.3	3.064
$\bar{k}_1$	1.693	1.773	4.5	1.601	5.4	2.038	20.3	1.601
$\bar{k}_2$	1.918	1.994	3.8	2.315	20.7	2.211	15.3	2.315
$\bar{k}_3$	1.816	1.693	7.3	1.613	11.2	1.416	22.0	0.740
$\bar{k}_4$	2.574	2.540	1.3	2.471	4.0	2.336	9.3	3.344
λ	1.028	1.016	1.2	1.091	6.1	1.128	9.7	1.091

Beispiel 5

$N = 4$, $K = 10$

$1/\mu_1 = 1.8$, $1/\mu_2 = 2.6$, $1/\mu_3 = 2.8$, $1/\mu_4 = 2.4$

$K_1 = 2$, $K_2 = 3$, $K_3 = 4$, $K_4 = 5$

	Sim	ZRTM	Abw	MVARLO	Abw	PROPIO	Abw	OVAP
$\bar{t}_1$	5.571	5.759	3.3	3.793	31.7	3.345	39.9	3.792
$\bar{t}_2$	7.668	9.136	16.1	9.334	17.8	6.620	15.8	9.334
$\bar{t}_3$	10.440	10.715	2.6	11.784	11.4	9.884	5.6	11.789
$\bar{t}_4$	12.880	11.188	15.1	7.424	42.3	12.286	4.8	7.424
$\bar{k}_1$	1.521	1.565	2.8	1.173	22.8	1.041	31.5	1.173
$\bar{k}_2$	2.098	2.483	15.5	2.886	27.7	2.060	1.8	2.886
$\bar{k}_3$	2.857	2.992	4.5	3.644	21.6	3.076	6.6	3.645
$\bar{k}_4$	3.524	3.041	15.9	2.297	34.8	3.823	7.8	2.296
λ	0.273	0.272	-	0.309	11.7	0.311	12.3	0.309

Beispiel 6

$N = 5$, $K = 10$

$1/\mu_1 = 1$, $1/\mu_2 = 2$, $1/\mu_3 = 1.5$, $1/\mu_4 = 1.8$, $1/\mu_5 = 1.6$

$K_i = 3$ für $i=1,2,3,4,5$

	Sim	ZRTM	Abw	MVARLO	Abw	PROPIO	Abw	OVAP
$\bar{t}_1$	4.718	4.431	6.5	3.685	21.9	3.212	31.9	1.685
$\bar{t}_2$	6.496	6.533	-	5.928	8.7	5.224	19.6	7.929
$\bar{t}_3$	4.498	4.871	7.7	3.676	18.3	4.531	-	3.676
$\bar{t}_4$	4.774	5.317	10.2	5.807	21.6	4.997	4.5	5.806
$\bar{t}_5$	4.558	4.678	2.6	4.278	6.1	4.707	3.3	4.278
$\bar{k}_1$	1.885	1.715	8.9	1.577	16.4	1.417	24.8	0.721
$\bar{k}_2$	2.595	2.529	2.5	2.536	2.3	2.304	11.2	3.392
$\bar{k}_3$	1.795	1.886	4.8	1.573	12.4	1.999	11.3	1.573
$\bar{k}_4$	1.906	2.059	7.4	2.484	30.4	2.204	15.6	2.484
$\bar{k}_5$	1.820	1.811	-	1.830	-	2.076	14.1	1.830
λ	0.399	0.387	3.1	0.428	6.8	0.441	10.3	0.428

Beispiel 7

$N = 5$, $K = 8$

$1/\mu_1 = 1$, $1/\mu_2 = 2$, $1/\mu_3 = 1.5$, $1/\mu_4 = 1.8$, $1/\mu_5 = 1.6$

$M_1 = 2$ $M_2 = 4$ $M_3 = 3$ $M_4 = 4$ $M_5 = 2$

	Sim	ZRTE	Abw	WVAPLO	Abw	PROBLO	Abw	OKAP
$\bar{t}_1$	2.046	2.159	5.5	1.598	21.9	1.701	16.0	1.598
$\bar{t}_2$	5.937	6.144	3.5	6.338	6.8	6.393	7.6	6.338
$\bar{t}_3$	3.667	3.593	1.0	3.274	10.7	3.537	3.5	3.274
$\bar{t}_4$	6.084	5.710	6.1	4.878	19.8	5.781	4.9	4.878
$\bar{t}_5$	2.873	3.058	6.5	3.744	30.3	2.501	12.0	3.745
$\bar{k}_1$	0.794	0.836	5.3	0.645	18.8	0.683	14.0	0.645
$\bar{k}_2$	2.306	2.379	3.2	2.557	10.9	2.569	11.0	2.557
$\bar{k}_3$	1.425	1.391	2.4	1.321	7.3	1.421	-	1.321
$\bar{k}_4$	2.361	2.210	6.4	1.968	16.7	2.323	1.6	1.968
$\bar{k}_5$	1.115	1.184	6.2	1.510	35.5	1.005	9.8	1.511
λ	0.388	0.387	-	0.403	3.9	0.402	3.5	0.403

Beispiel 8

$N = 3$, $K = 10$

$1/\mu_1 = 0.2$, $1/\mu_2 = 1.2$, $1/\mu_3 = 1.4$

$M_1 = 8$, $M_2 = 7$, $M_3 = 6$, $p_{12} = p_{13} = 0.5$ (CSM)

$p_{1j}=0.5$ für $j=2,3$ CSM= Central-Server-Model

	Sim	ZRTE	Abw	WVAPLO	Abw	PROBLO	Abw	OKAP
$\bar{t}_1$	1.371	1.415	3.0	0.974	29.0	0.415	69.7	0.274
$\bar{t}_2$	5.199	4.739	8.8	5.007	3.7	7.009	25.8	5.008
$\bar{t}_3$	6.874	7.162	4.0	7.640	10.0	6.351	7.6	9.039
$\bar{k}_1$	1.850	1.921	3.6	1.335	27.8	0.585	68.3	0.376
$\bar{k}_2$	3.480	3.217	7.5	3.431	1.4	4.939	29.5	3.433
$\bar{k}_3$	4.670	4.862	3.9	5.234	10.7	4.475	4.2	6.193
λ	1.349	1.358	-	1.370	1.6	1.409	4.5	1.370

Beispiel 9

$N = 4$, $K = 15$

$1/\mu_1 = 1$, $1/\mu_2 = 0.9$, $1/\mu_3 = 0.667$, $1/\mu_4 = 0.8$

$N_1 = 3$, $N_2 = 6$, $N_3 = 7$, $N_4 = 4$

$p_{12} = 0.6$, $p_{21} = 0.55$, $p_{32} = 0.5$, $p_{43} = 1.0$

$p_{13} = 0.4$, $p_{24} = 0.45$, $p_{34} = 0.5$

	Sim	ZRTM	Abw	MVARIO	Abw	PROBIO	Abw	OKAP
$\bar{t}_1$	3.141	3.115	-	2.265	27.8	2.598	20.8	2.095
$\bar{t}_2$	5.331	5.375	-	5.506	-	4.970	11.2	4.634
$\bar{t}_3$	4.364	4.891	10.1	4.239	2.8	4.461	2.1	3.495
$\bar{t}_4$	3.335	3.286	1.4	2.743	17.7	2.650	25.8	4.234
$\bar{k}_1$	1.539	1.483	3.6	1.212	21.2	1.397	9.2	1.122
$\bar{k}_2$	4.923	4.654	5.4	5.359	8.8	4.858	1.3	4.705
$\bar{k}_3$	5.210	5.675	8.1	5.529	6.1	5.844	10.8	4.558
$\bar{k}_4$	3.328	3.187	4.2	2.991	10.1	2.901	12.8	4.616
λ	0.492	0.476	2.8	0.535	9.2	0.538	9.2	0.535

Beispiel 10

$N = 4$, $K = 14$

$1/\mu_1 = 15.5$, $1/\mu_2 = 21.36$, $1/\mu_3 = 18.16$, $1/\mu_4 = 14.68$

$N_i = 4$ (i=1,2,3,4)

$p_{ij} = 0.25$ (i,j=1,2,3,4)

	Sim	ZRTM	Abw	MVARIO	Abw	PROBIO	Abw	OKAP
$\bar{t}_1$	100.11	89.009	12.4	134.818	25.8	73.186	26.8	46.229
$\bar{t}_2$	95.87	102.913	7.3	158.823	39.6	76.958	24.5	151.691
$\bar{t}_3$	88.18	95.553	8.3	136.966	35.6	75.237	17.2	76.209
$\bar{t}_4$	89.01	86.886	2.3	127.360	30.1	72.370	22.9	39.941
$\bar{k}_1$	3.65	3.328	8.8	3.383	7.3	3.442	5.7	2.061
$\bar{k}_2$	3.49	3.848	10.2	3.985	12.4	3.618	3.5	6.762
$\bar{k}_3$	3.21	3.573	11.3	3.437	6.6	3.537	9.2	3.397
$\bar{k}_4$	3.24	3.249	-	3.196	1.3	3.403	4.8	1.780
λ	0.036	0.037	2.7	0.025	45.6	0.047	18.4	0.446

SYMBOLVERZEICHNIS

AFWS	Ablaufwarteschlange
c_i	Variationskoeffizient der Bedienzeit der Station i
e_i	Mittlere Besuchshäufigkeit, die ein Auftrag bei Station i macht.
G	Normalisierungskonstante
K	Gesamtanzahl der Aufträge im Netz
$\bar{k}_i$	Mittlere Anzahl von Aufträgen in der i-ten Station (wartend + in Bedienung)
M_i	Kapazität der Station i
KWS	Kommunikationswarteschlange
m_i	Anzahl der Bedieneinheiten in der i-ten Station
N	Gesamtanzahl der Stationen im Netz
[P]	Matrix der Übergangswahrscheinlichkeiten
p_{ij}	Wahrscheinlichkeit, daß ein Auftrag nach Beendigung in der Station i anschl. zur Station j gelangt.
$p_i(k_i)$	Randwahrscheinlichkeiten
$p_i(l/k)$	Wahrscheinlichkeit, daß in der Station l Aufträge befinden, unter der Bedingung, daß k Aufträge im Gesamtsystem sind.
P_{B_i}	Blockierwahrscheinlichkeit der Station i
Q_i	Mittlere Warteschlangenlänge der Station i
R	Gesamtanzahl der Klassen
r	Eine Auftragsklasse
t	Gesamtverweilzeit des Systems
$\bar{t}_i$	Mittlere Verweilzeit eines Auftrags in Station i
$\bar{t}_{B_i}$	Mittlere Blockierzeit der Station i
x_i	Relative Auslastung der Station i
λ	Gesamtdurchsatz des Systems
λ_i	Durchsatz der i-ten Station
μ_i	Mittlere Bedienrate eines Auftrags in Station i
ρ_i	Auslastung der i-ten Station
⊗	Faltung

LITERATURVERZEICHNIS

- /AKYL 81/ Akyildiz I. F. ; Bolch G.:
Analytic Solution Techniques for Queueing Network
Models of Computer Systems,
Arbeitsbericht des IMMD der Universität Erlangen-
Nürnberg, Band 14, Nr. 4, 1981
- /AKYL 82/ Akyildiz I. F. ; Bolch G.:
Möglichkeiten und Grenzen der analytischen
Modellbildung von Warteschlangensystemen
Informatik Fachberichte, (edit. M. Goller),
Springer Verlag, Herbst 1982.
- /AKYL 83/ Akyildiz I. F. ; Bolch G.:
Erweiterung der Mittelwertanalyse zur Berechnung
der Zustandswahrscheinlichkeiten für geschlossene
und gemischte Netze
2. Int. Tagung für Messung, Modellierung und
Bewertung von Rechensystemen, Stuttgart, 1983
- /AKYL 84/ Akyildiz I. F. ; Bolch G.:
Analytical Methods for Computer Systems Performance
Modelling
AMSE Simulation and Modelling, Athens, 1984
- /AKYL 84/ Akyildiz I. F. ; Bolch G.:
A Storage and Run-time Reducing Method for Queueing
Network Models of Computer Systems
AMSE Simulation and Modelling, Athens, 1984
- /AKYL 84/ Akyildiz I. F. ; Blümle A. ; Ruf T.:
Hybrid-Simulation eines Multiprozessorsystems mit
Prozeßsynchronisation
ASIM Simulationstechnik, Wien, 1984

- /BALS 82/ Balsamo S. ; Iazeolla G.:
An Extension of Norton's Theorem for Queueing Networks
IEEE Transactions on Software Engineering
Vol. SE-8, Nr. 4, July 1982, S. 298-305
- /BARD 79/ Bard Y.:
Some Extension to Multiclass Queueing Network Analysis
Fourth Int. Symposium on Modelling and Performance Evaluation of Computer Systems, Feb.1979, Vol.1, Vienna
- /BCMP 75/ Baskett F., Chandy K., Muntz R., Palacios G:
Open, Closed and Mixed Network of Queues with Different Classes of Customers
JACM, Vol.22, 2, Apr. 1975, pp. 248-260
- /BLRF 84/ Blümle A. ; Ruf T.:
Kopplung von Software- und Hardware Modellen zur Leistungsanalyse von Multiprozessorsystemen mit Prozeßsynchronisation
Studienarbeit am IMMD IV der FAU Erlangen-Nürnberg, 1984
- /BOAK 82/ Bolch G. ; Akyildiz I. F.:
Analyse von Rechensystemen
Teubner Verlag, 1982
- /BOXM 81/ Boxma O. J. ; Konheim A. G.:
Approximate Analysis Exponential Queueing Systems with Blocking
Acta Informatica 15, January 1981, pp. 19-66
- /BUZE 71/ Buzen J. P.:
Queueing Network Models of Multiprogramming
Ph. D. Thesis, Div. Eng. and Applied Science, Harvard Univ., Cambridge, Mass., Aug. 1971

- /BUZE 73/ Buzen J. P.:
Computational Algorithms for Closed Queueing
Networks with Exponential Servers
CACM 16, 9, Sept. 1973, pp. 527-531
- /BUZE 76/ Buzen J. P.:
Fundamental Operational Laws of Computer System
Performance
Acta Informatica 7, 2, 1976, pp. 167-182
- /CHAN 72/ Chandy K. M.:
The Analysis and Solutions for General Queueing
Networks
Proc. of the Sixth Annual Princeton Conf. on
Information Sciences and Systems, Princeton Univ.,
March 1972, pp.224-228
- /CHAN 75a/ Chandy K., Herzog U., Woo L.:
Parametric Analysis of Queueing Network Models
IBM Journal Res. Dev. 19, 1, Jan. 1975, pp.36-42
- /CHAN 75b/ Chandy K., Herzog U., Woo L.:
Approximate Analysis of General Queueing Networks
IBM Journal Res. Dev. 19, 1, Jan. 1975, pp.43-49
- /CHAN 76/ Chandy K., Howard J.H., Towsley D.:
Product Form and Local Balance in Queueing Networks
Modelling and Performance Evaluation of Computer
Systems, Workshop Preprints ed. Gelenbe and Beil-
ner, North-Holland Publishing Co., Oct. 1976,
pp89-103
- /CHAN 83/ Chandy K. ; Martin J.:
A Characterization of Product-Form Queueing
Networks
JACM, Vol. 30, Nr 2, April 1983, pp.286-299

- /COUR 77/ Courtois P. J.:
Decomposability: Queueing and Computer System
Applications
Academic Press, New York, 1977
- /DENN 78/ Denning P. J. ; Buzen J. P.:
Operational Analysis of Queueing Networks
Computing Surveys 10, 3, Sept. 1978, pp.225-261
- /FELD 83/ Feldmann M.:
Implementierung von Verfahren zur Leistungsanalyse
von Rechensystemen, Teil 3: SCAT
Studienarbeit am IMMD IV der FAU Erlangen-Nürnberg,
1983
- /GELE 75/ Gelenbe E.:
On Approximate Computer System Models
JACM, Vol. 22, 2, Apr. 1975, pp.261-269
- /GEOR 81/ Georges J.:
Amendments to an Analysis Method for Queueing
Networks with Blocking
Technical Report, Philips, 1981
- /GORD 67/ Gordon W. ; Newell G.:
Closed Queueing Systems with Exponential Servers
Operations Res. 15, 1967, pp.254-265
- /GROS 84/ Großer B.:
Algorithmen zur Leistungsanalyse von Rechensystemen
Studienarbeit am IMMD IV der FAU Erlangen-Nürnberg,
1984
- /HERZ 75/ Herzog U., Woo L., Chandy K.:
Solution of Queueing Problems by a Recursive
Technique
IBM Journal Res. Dev. 19, May 1975, pp.295-300

- /HERZ 83/ Herzog U.:
Gespräch in der Mensa, Dez. 1983
- /HORD 81/ Hordijk A. ; van Dijk N.:
Networks of Queues with Blocking
The 8-th Int. Symp. on Computer Performance
Modelling, Measurement and Evaluation, Nov. 4-6,
1981, Amsterdam
- /JACK 57/ Jackson J.:
Networks on Waiting Lines
Operations Res. 5, 1957, pp. 518-521
- /JACK 63/ Jackson J.:
Jobshop-like Queueing Systems
Management Science 10, 1, 1963, pp.131-142
- /JACO 82/ Jacobson P. A. ; Lazowska E. D.:
Analyzing Queueing Networks with Simultaneous
Resource Possession
CACM, Vol. 25, Nr.2, Feb. 1982, pp.142-151
- /KELL 78/ Kelly F. P.:
Reversibility and Stochastic Networks
John Wiley, New York, 1978
- /KOBA 74/ Kobayashi H.:
Application of the Diffusion Approximation to
Queueing Networks, Part I: Equilibrium Queue Distri-
butions
JACM, Vol.21, 2, Apr. 1974, pp.316-328
- Application of the Diffusion Approximation to
Queueing Networks, Part II: Nonequilibrium Distri-
butions and Computer Modelling
JACM, Vol.21, Nr.3, July 1974, pp. 459-469

- /KONH 76/ Konheim A. G. ; Reiser M.:
A Queueing Model with Finite Waiting Room and
Blocking
JACM, Vol. 23, Nr. 2, Apr.1976, pp.328-341
- /KONH 77/ Konheim A. G. ; Reiser M.:
Finite Capacity Queueing Systems with Applications
in Computer Modelling,
SIAM J. Computing, Vol.7, Nr.2, May 1977, pp.210-
229
- /KRIT 82/ Kritzinger P. S. ; van Wyk S. ; Krzesinski A. E.:
A Generalization of Norton's Theorem for Multiclass
Queueing Networks
Performance Evaluation 2, 1982, pp.98-107
- /KÜHN 79/ Kühn P. J.:
Approximate Analysis of General Queueing Networks
by Decomposition
IEEE Transactions on Communications, Vol. Com-27,
Nr.1, Jan. 1979, pp.113-126
- /LABE 80/ Labetoulle J. ; Pujolle G.:
Isolation Method in a Network of Queues
IEEE Transactions on Software Engineering, Vol. SE-
6, Nr.4, July 1980
- /LAVE 83/ Lavenberg S. S.:
Computer Performance Modelling Handbook
Academic Press, New York, 1983
- /MARI 80/ Marie R.:
Calculating Equilibrium Probabilities for
(n)/C/1/N Queues
ACM Sigmetrics, Performance Eval. Review, Vol. 9,
Nr.2, Summer 1980, pp.117-125

- /MUNT 73/ Muntz R. R.:
Poisson Departure Process and Queueing Networks
Proc. of the 7th Ann. Princeton Conf. on Inf.
Sciences and System, Princeton Univ., Princeton,
N. J., March 1973, pp.435-440
- /MUNT 74/ Muntz R. R. ; Wong J. W.:
Asymptotic Properties of Closed Queueing Network
Models
Proc. of the 8th Ann. Princeton Conf. on Inf. Sci-
ences and Systems, Princeton Univ., N.J., March
1984, pp. 348-352
- /NEUS 81/ Neuse D. ; Chandu K.:
SCAT: A Heuristic Algorithm for Queueing Network
Models of Computing Systems
ACM Sigmetrics, Vol. 10, Nr.3, pp.59-79, Sept.1981
- /NILS 81/ Nilsson A. ; Altiok T.:
Open Queueing Networks with Finite Capacity Queues
IEEE Transactions on Software Eng., 1981, pp.87-91
- /NOET 79/ Noetzel A. S.:
A Generalized Queueing Discipline for Product Form
Network Solutions
JACM, Vol.26, Nr.4, Oct.1979, pp.779-793
- /PERR 81/ Perros H. G.:
A Symmetrical Exponential Open Queue Network with
Blocking and Feedback
IEEE Transactions on Software Eng., Vol. SE-7,
Nr.4, July 1981
- /PITT 79/ Pittel B.:
Closed Exponential Networks of Queues with
Saturation: The Jackson Type Stationary Distribu-
tion and its Asymptotic Analysis
Math. of Operations Res., Vol. 4, 1979, pp. 367-378

- /PATE 84a/ Paterok M.:
Erweiterte Parametrische Analyse, (EPA)
Studienarbeit am IMMD IV der FAU Erlangen-Nürnberg,
1984
- /PATE 84b/ Paterok M.:
QUEBLO - ein Programmpaket zur Analyse von Blok-
kiernetzen
Diplomarbeit am IMMD IV der FAU Erlangen-Nürnberg,
1984
- /REIS 74/ Reiser M. ; Kobayashi H.:
Accuracy of the Diffusion Approximation for Some
Queueing Systems,
IBM Journal Res. Dev. 18,2,March 1974, pp. 110-124.
- /REIS 75/ Reiser M. ; Kobayashi H.:
Queueing Networks with Multiple Closed Chains:
Theory and Computational Algorithms
IBM Journal Res. Dev. 19, May 1975, pp.283-294
- /REIS 79/ Reiser M.:
Mean Value Analysis of Queueing Networks, A New
Look at an Old Problem
Fourth Int. Symp. on Modelling and Performance
Eval. of Computer Systems, Vol.I, Feb. 1979, Vienna
- /REIS 80/ Reiser M. ; Lavenberg S. S.:
Mean Value Analysis of Closed Multichain Queueing
Networks
JACM, Vol. 27, Nr.2, Apr. 1980, pp.313-322
- /REIS 81/ Reiser M.:
Mean Value Analysis and Convolution Method for
Queueing Dependent Servers in Closed Queueing Net-
works
Performance Evaluation, Vol.I, Nr.1, Jan.1981,pp.7-
18

- /SAUE 81/ Sauer C. ; Chandy K.:
Computer Systems Performance Modelling
Prentice Hall, Englewood Cliffs, N. J., 1981
- /SCHM 84/ Schmidt B.:
GPSS - Fortran Version 3
RRZE-Dokumentation, Bd. 119, Erlangen
- /SCHW 79/ Schweitzer P.:
Approximate Analysis of Multiclass Closed Networks
of Queues
Int. Conf. Stochastic Control and Optimization,
Amsterdam, 1979.
- /SEVC 81/ Sevcik K. ; Mitrani I.:
The Distribution of Queueing Network States at
Input and Output Instants
JACM, Vol.28, Nr2, Apr.1981, pp.358-371
- /SHUM 76/ Shum A.W.C.:
Queueing Models for Computer Systems with General
Service time Distributions
PhD Thesis, Div. Eng. and Applied Physics, Mass.,
Dec. 1976
- /SPIR 79/ Spirn J. P.:
Queueing Networks with Random Selection for Service
IEEE, T-SE 5, 1979, pp.287-289
- /STEW 78/ Stewart W. J.:
A Comparison of Numerical Techniques in Markov
Modelling
CACM, Vol. 21, Nr.2, Feb.1978, pp.144-152
- /TAKA 80/ Takahashi Y., Miyahara H., Hasegawa T.:
An Approximation Method for Open Restricted
Queueing Networks
Operations Research, 28, 1980, pp.594-602

/WONG 75/

Wong J. W.:

Queueing Network Models for Computer Systems

PhD Thesis, UCLA-Eng-7579, June 1975

/ZAHO 81/

Zahorjan J., Sevcik K., Eager D., Galler B.:

Balanced Job Bound Analysis of Queueing Networks

CACM 25, 2, 1981, pp. 134-141

Lebenslauf

Name	Ian Fuat Akyildiz
Geboren	11.04.1954 in Istanbul
Nationalität	Türkisch

Ausbildung

1960 - 1965	Grundschule in Istanbul
1965 - 1969	Unterstufe des österreichischen Sankt Georg Kollegs in Istanbul
1965 - 1974	Oberstufe des österreichischen Sankt Georg Kollegs in Istanbul
1974 - 1975	Studienkolleg bei den wiss. Hochschulen in München
WS 1975/76	Beginn des Studiums der Informatik an der Universität Erlangen-Nürnberg
April 1978	Vordiplom
Oktober 1980	Hauptdiplomprüfungen
November 1981	Diplom
Oktober 1982	Veröffentlichung eines Buches
September 1984	Dissertation

Beruf

1981 - jetzt	wiss. Mitarbeiter am Lehrstuhl für Betriebssysteme an der Univ. Erlangen-Nürnberg
1983 - jetzt	freier Mitarbeiter an der Grundig Akademie
1983 - jetzt	freier Mitarbeiter an der University of Maryland European Division

In letzter Zeit sind folgende Arbeitsberichte des Instituts für
Mathematische Maschinen und Datenverarbeitung erschienen:

=====

Band 13

=====

- Nr. 1 Schrammel, D., Bayer, W., Gliemann, H.-P., Heigl, M.,
Mertens, P.:
Ein Leistungs- und Wirtschaftlichkeitsvergleich zwischen
Klein- und Großcomputern
(Januar 1980)
- Nr. 2 Göttler, H.:
Semantische Modelle
(März 1980)
- Nr. 3 Fromm, H.J.:
Zur Modellierung der Speicherinterferenz bei hierarchisch
organisierten Multiprozessorsystemen
(April 1980)
- Nr. 4 Jäpel, D.:
Klassifikatorbezogene Merkmalsauswahl
(Juli 1980)
- Nr. 5 Bodendorf, F., Mertens, P., Haas, R., Weber, G., Wolf, G.:
SAMBA - ein Methodenbankrahmen um das Statistik-Paket SPSS
(Juli 1980)
- Nr. 6 Bartsch, B.:
Inferenz und Analyse spezieller Graphgrammatiken für die
syntaktische Mustererkennung
(September 1980)
- Nr. 7 Wurm, F.X.:
Auftragssystem für eine Mehrprozessoranlage
(Oktober 1980)
- Nr. 8 Schedelbeck, A.:
Vergleich der beiden höheren Realzeitprogrammiersprachen
ADA und PEARL
(Oktober 1980)
- Nr. 9 Jahresbericht 1980 der Informatik
(März 1981)

- Nr. 1 Pattern Recognition
 (Research at Lehrstuhl für Informatik 5)
 (Februar 1981)
- Nr. 2 Bunke, H.:
 Analyse elektrischer Schaltpläne
 (März 1981)
- Nr. 3 Linster, C.-U.:
 SYMPOS/UNIX - Ein Betriebssystem für homogene
 Polyprozessorsysteme
 (Juni 1981)
- Nr. 4 Akyildiz, J.F., Bolch, G.:
 Analytic Solution Techniques for Queueing Network
 Models of Computer Systems
 (Juli 1981)
- Nr. 5 Lindstedt, W.G.:
 Ein Verfahren zur Erstellung portabler algorithmischer
 Laufzeitprogramme
 (Juli 1981)
- Nr. 6 Vollmar, R.:
 Zum Begriff des Parallelismus bei Polyautomaten
 (Juli 1981)
- Nr. 7 Luft, A.L.:
 Rationaler Sprachgebrauch und orthosprachliche
 Standardisierung als Grundlagen zuverlässiger Software-
 Entwicklung und -Dokumentation
 (Januar 1982)
- Nr. 8 Händler, W.:
 PROCEEDINGS OF A WORKSHOP ON TAXONOMY IN COMPUTER
 ARCHITECTURE
 (Februar 1982)
- Nr. 9 Beth, Th., Hess, P., Wirl, K.:
 Materialien zur Kryptographie
 (März 1982)
- Nr. 10 Jahresbericht 1981 der Informatik
 (März 1982)

- Nr. 1 Wittmann, A.:
Ein Mechanismus für die Synchronisation paralleler
Prozesse
(Februar 1982)
- Nr. 2 Maehle, E.:
Fehlertolerantes Verhalten in Multiprozessoren -
Untersuchungen zur Diagnose und Rekonfiguration
(März 1982)
- Nr. 3 Bathelt, P.:
Vergleich von Synchronisationsmechanismen
(Mai 1982)
- Nr. 4 Keramidis, S.:
Eine Methode zur Spezifikation und korrekten
Implementierung von asynchronen Systemen
(Juni 1982)
- Nr. 5 Fromm, H.J.:
Multiprozessor-Rechenanlagen:
Programmstrukturen, Maschinenstrukturen und Zuordnungs-
probleme
(Mai 1982)
- Nr. 6 Bley, H.:
Vorverarbeitung und Segmentierung von Stromlaufplänen
unter Verwendung von Bildgraphen
(Juli 1982)
- Nr. 7 Hchl, W.:
Informatik-Sammlung
Katalog mit historischen Erläuterungen
(August 1982)
- Nr. 8 Messerer, M.:
Ein neuer Ansatz zur Parallelisierung von
Compilern
(August 1982)
- Nr. 9 Grosch, J.:
Eine Programmiersprache mit mengentheoretischen
Konstrukten und deren effiziente Implementierung
(August 1982)
- Nr. 10 Kleinöder, W.:
Stochastische Bewertung von Aufgabenstrukturen
für hierarchische Mehrrechnersysteme
(August 1982)

- Nr. 11 Kneißl, F.:
Realisierung von Makro-Datenflußmechanismen auf
hierarchischen Mehrrechnersystemen
(August 1982)
- Nr. 12 Brendel, W., Fan, Z., Klar, R., Schmielau, W.:
ERES 82
Handbuch und Fallstudie
Handbook and Case Study
(Oktober 1982)
- Nr. 13 Pelz, K.:
Portable Codegenerierung in Übersetzern
für Prozeß-Programmiersprachen
(Oktober 1982)
- Nr. 14 Sommer, U., Steinchen, J.:
Ein Kleinrechner-Grafiksystem zur Dokumentation
von EDV-Abläufen
(Oktober 1982)
- Nr. 15 Hein, H.-W.:
Das Erlanger Spracherkennungssystem
(November 1982)
- Nr. 16 Bolch, G., Bruchner, W.:
Analytische Modelle für symmetrische Mehrprozessor-
anlagen mit dynamischen Prioritäten
(Dezember 1982)
- Nr. 17 Jahresbericht 1982 der Informatik
(März 1983)

- Nr. 1 Gall, R.:
Formale Beschreibung des inkrementellen Programmierens-
im-Großen mit Graph-Grammatiken
(März 1983)
- Nr. 2 König, R.:
Beiträge zur algebraischen Theorie der formalen Sprachen
(März 1983)
- Nr. 3 Welbourn, D.B.:
The Design of Mechanical Components and the Development
of DUCT:
17 Years of CAD/CAM in Cambridge University
(April 1983)
- Nr. 4 Woitok, R.:
Die System-Implementierungs-Sprache ALICE
(Mai 1983)
- Nr. 5 Schneider, S., Schwab, P., Renninger, W.:
Wesen, Vergleich und Stand von Software zur Produktion
von Systemen der computergestützten Unternehmensplanung
(Juli 1983)
- Nr. 6 Händler, W., Herzog, U., Hofmann, F., Schneider, H.J.
und Mitarbeiter:
Projektabschlußbericht 1978 - 1982 Erlangen General Purpose
Array
(Juli 1983)
- Nr. 7 Mackert, L.:
Modellierung, Spezifikation und korrekte Realisierung
von asynchronen Systemen
(Juli 1983)
- Nr. 8 Reitenspieß, M.:
Sprachkonstrukte zur Spezifikation und korrekten Implemen-
tierung von Schutzproblemen
(Juli 1983)
- Nr. 9 Weber, K.:
Modellierung von Fehlverhalten mit Berücksichtigung
paralleler Abläufe
(August 1983)
- Nr. 10 Dietsch, H.:
Ein verteiltes Kommunikationssystem für einen Verbund
aus lokalen Mikrorechner-Netzwerken
(Dezember 1983)

- Nr. 11 Maehle, E., Schmitter, E.:
Workshop Fehlertolerante Mehrprozessor- und Mehr-
rechnersysteme
(Dezember 1983)
- Nr. 12 Volkert, J.:
Multiplikative Abstandsfunktionen und deren Anwendung
in der Bildverarbeitung
(Dezember 1983)
- Nr. 13 Steinbauer, D.:
Transaktionen als Grundlage zur Strukturierung und
Integritätssicherung in Datenbank-Anwendungssystemen
(Dezember 1983)
- Nr. 14 Wilke, P.:
Zusammenhänge und Unterschiede zwischen Graph-Grammatiken
und Petri-Netzen sowie verwandter Systeme
(Dezember 1983)
- Nr. 15 Jahresbericht der Informatik 1983
(März 1984)

- Nr. 1 Eberlein, W.:
Architektur technischer Datenbanken für
integrierte Ingenieursysteme
(Februar 1984)
- Nr. 2 Mertens, P., Heigl, M.:
Neuere Entwicklungen der computergestützten
Produktionsplanung, Eignung - Verbindungen -
Entwicklungspfade
(April 1984)
- Nr. 3 Stoyan, H.:
Maschinen- unabhängige Code-Erzeugung als semantik-
erhaltende beweisbare Programmtransformation
(Juli 1984)
- Nr. 4 Brendel, W.:
Funktionaler Entwurf und automatische Synthese
hochintegrierter Schaltkreise
(August 1984)
- Nr. 5 Brietzmann, A.:
Semantische und pragmatische Analyse im Erlanger
Spracherkennungsprojekt
(Oktober 1984)
- Nr. 6 Hoppe, B.:
Deterministisches Zuordnen vielfach durchlaufener
Aufgabensysteme in Multiprozessorsystemen
(Oktober 1984)
- Nr. 7 Fumy, W., Posch, S., Rieß, H.P.:
Materialien zur Codierungstheorie III
(Dezember 1984)

